
AMOS
THE CREATOR
HAND BUCH
europress
SOFTWARE

AMOS

The Creator

(DER SCHÖPFER)

von Francois Lionet

Europress/Jawx 1992

europress SOFTWARE



Design und Programmierung	Francois Lionet
Projektmanagement	Richard Vanner
Verfasser des Handbuchs	Stephen Hill
Druckvorlage von	Richard Vanner und Visual Eyes
Übersetzung	Salford Translations Ltd Carsten Bernhard

AMOS Verpackung von VisualEyes

Hilfe bei defekten Disketten und anderen Einstiegsproblemen erhalten Sie von Customer Services (Kundendienst), Mandarin Software, Europa House, Adlington Park, Adlington, Macclesfield, Cheshire SK10 4NP, England.

ISBN# 9781872084509 Erstausgabe Juni 1990 Überarbeitet Februar 1992

Und wie kam all dies zustande?

AMOS-Basic wurde von Francois Lionet entwickelt und programmiert. Aufgrund seiner schlauen Ideen und kreativen Einfälle entstand wohl die beste Programmiersprache, die gegenwärtig für den AMIGA verfügbar ist.

AMOS wurde unter Einsatz der folgenden Programme entwickelt:

Devpac II Assembler - HiSoft
Deluxe Paint III - Electronic Arts
Pix Mate - Progressive Peripherals & Software
Cross-Dos - Consultron
Mini Office Professional Communications - Database Software

EuroPress Software bedankt sich für die freundliche Hilfe bei der Entwicklung von AMOS bei:

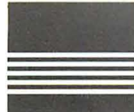
Allistair Brimble, Aaron und Adam Fothergill von Shadow Software, Peter Hickman, Rico Holmes, Commodore UK für die internationalen Tastatur-Layouts (und den Amiga), Commodore Frankreich für die Hilfe bei der A-1000-, CPTV- und A300- Problemstellung, 17-Bit Software für Samples und Demos, Martyn Brown für die Hilfe bei der Erstellung einer super PD-Library, Virus Free PD für Soundtracker, Simon Cook für seine konstruktiven Kommentare und die Suche nach Bugs, Lee, Alex und allen anderen AMOS-Entwicklern für ihre freundliche Unterstützung und letztendlich auch Ihnen allen dafür, daß Sie so geduldig auf diese Software gewartet haben. Wir hoffen, daß auch Sie finden, daß sich das Warten gelohnt hat.

Dieses Handbuch wurde mit WriteNow und das Layout mit Page Maker auf dem Apple Macintosh erstellt.

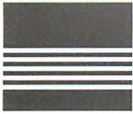
Copyright

Mit AMOS können Sie Spitzensoftware programmieren, deshalb ist es wichtig, daß Sie AMOS in Ihren Programmen erwähnen und z.B. sagen "Programmiert von Hans Müller mit AMOS". Wenn möglich, sollten Sie auch das AMOS-Sprite einsetzen.

Wenn Sie Ihr Progamm vermarkten, so muß "AMOS 1990 Europress Software" auf der Rückseite der Verpackung und den gedruckten Anleitungen angegeben werden.



Inhaltsverzeichnis



1: Einleitung

Willkommen in der aufregenden Welt von AMOS - Dem Schöpfer! Sie wissen ja, was für ein fantastischer Computer der Amiga ist, und jetzt können Sie endlich seine Kapazität voll ausschöpfen.

Im September 1988 brachte EuroPress Software STOS Basic für den ST auf den Markt. Damit gingen wir in die Geschichte ein, denn das war die erste Programmiersprache, die in der Hitliste für ST-Spiele die absolute Nummer Eins war. Und jetzt wurde STOS von Grund auf neu geschrieben, und so ist AMOS- Basic für den Amiga entstanden. AMOS-Basic bietet eine unwahrscheinliche Vielzahl an Befehlen - über 500, um genau zu sein - die größtenteils wahnsinnig leistungsstark sind. Zum Beispiel können Sie mit nur einem Basic-Befehl einen Schirm hüpfen lassen oder einen Sprite animieren.

AMOS ist mehr als nur eine neue Basic-Version. Es ist vielmehr ein spezielles Programm zum Schreiben von Spielen und verfügt über seine eigene, eingebaute Animationssprache (AMAL). Ein starkes Unterbrechersystem (Interrupt) sorgt dafür, daß AMAL Programme fünfzigmal pro Sekunde ablaufen. Damit kann man so ziemlich alles machen, angefangen von den Angriffswellen in einem Ballerspiel bis zu einem spiegelglatten Hardware-Scrolling-Effekt. Und gleichzeitig kann Ihr Basic- Programm noch etwas ganz anderes!

Es spielt gar keine Rolle, wieviel Sie vom Programmieren verstehen, bei AMOS liegen Sie immer richtig. Wenn Sie noch nie ein Spiel geschrieben haben, dann jagt Ihnen vielleicht die bloße Vorstellung schon kalte Schauer über den Rücken. Aber vergessen Sie nicht, daß viele der alten Klassiker eigentlich ganz einfache Programme mit ein oder zwei Besonderheiten sind - nehmen wir zum Beispiel nur einmal Tetris. Wie stark Ihr Spiel letztendlich ist hängt viel von Ihren guten Ideen und nicht nur von Ihren Programmierkünsten ab. Und dann hilft AMOS noch ein bißchen nach, und im Nu können auch Sie ganz professionelle Spiele programmieren. Alles, was Sie wirklich brauchen, ist Ihre Phantasie!

Wenn Sie dann ein Spiel in AMOS-Basic geschrieben haben, sollten Sie es aber nicht nur für sich behalten. Mandarin Software brennt darauf, Programme, die mit AMOS-Basic geschrieben werden, zu vertreiben. Da macht es auch nicht aus, wenn es beim Programmieren noch etwas geholptert hat, es kommt nur darauf an, daß Sie gute Ideen haben. Dann ist es gut möglich, daß Sie als Spieleschreiber eine Zukunft haben. Also schicken Sie uns bitte Ihre Spiele.

Außerdem würden wir uns freuen, wenn Sie uns Ihre Anregungen und Kommentare zusenden. So entstanden zum Beispiel einige der Eigenschaften von AMOS direkt aufgrund der Ideen von STOS-Anwendern. Richten Sie Ihre Briefe an Richard Vanner, Development Manager, Mandarin Software, Adlington Park, Adlington, Macclesfield SK10 4NP England.

Und lassen wir noch den Mann, der diese Software geschrieben hat, den unglaublich begabten Franzosen Francois Lionet zu Wort kommen. Francois - man glaubt es kaum - war eigentlich ein Tierarzt und hat das ursprüngliche STOS Basic mehr oder weniger so nebenher geschaffen.

Widmung von Francois Lionet

Ich widme dieses Programm meiner Frau Carine für ihre Geduld mit dem Programmierer-Streß!

Vorwort

Ich möchte Ihnen gerne versichern, daß ich all mein Wissen in dieses Produkt gesteckt habe, und jetzt bietet es alles, was ich immer von einem Basic für den Amiga erwartet habe.

Wenn Sie ein Spiel programmieren, dann gehen mehr als drei Viertel der Zeit dafür drauf, daß Sie grundlegende Routinen für die Darstellung von Sprites schreiben, Schirme zu bewegen usw. Das ist auf jedem Computer ein hartes Brot, aber mit dem Amiga ist es ein wirklicher Alptraum, weil er so leistungsstark ist. In den Jahren vor Christus - nein Entschuldigung, vor AMOS - bedeutete das Schreiben eines Spiels auf dem Amiga einen großen Zeitaufwand für Utilities.

Bei AMOS habe ich alles schon für Sie vorbereitet, deshalb brauchen Sie sich nicht mehr mit den langweiligen BOB-Routinen, Berechnungen der Copper-Listen, dem Jonglieren von Disketten usw. aufzuhalten. Sie können sich auf das wirklich Wichtige beim Schreiben von Spielen konzentrieren: das Thema, die Grafik und die Musik.

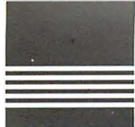
Ich hoffe, daß Sie mit AMOS Ihre Ideen verwirklichen können. Und wenn mir in ein paar Jahren der Programmieren von einem Hit erzählt, daß er das Spiel mit Hilfe von AMOS geschrieben hat, dann werde ich mich wahnsinnig freuen.

AMOS ist anpassungs- und ausbaufähig, und kann so mit der Entwicklung Ihres geliebten Computers mitwachsen. Das sieht man am besten in den Musik-Routinen. Die sind nämlich Public Domain, und den Quelltext (in der Assembler- Programmiersprache) finden Sie auf der AMOS-Programmdiskette. Warum? Weil sie das beste Beispiel dafür sind, wie man für AMOS-Basic Erweiterungen schafft. Mit der gleichen Technik kann man neue Anweisungen hinzufügen, oder die Musik- Routine auf den Stand der neuesten 235-Stimmen-Sound-Tracker-Version 4.242 Rev 1.2 bringen.

Zum Schluß noch ein Wort - Sie haben es erraten - zum Thema Raubkopieren. Raubkopieren ist wirklich ein Verbrechen, ein schleichendes Verbrechen. Eine Auswirkung ist der Boom auf dem Markt der Spiel-Konsolen. Da man Module einfach nicht kopieren kann, gehen jetzt alle Software-Häuser dazu über, sie einzusetzen. Aber wenn Sie in einigen Jahren noch einen erschwinglichen Computer mit einem Laufwerk und einer Tastatur finden wollen, dann sollten Sie mich unterstützen. Also kopieren Sie AMOS bitte nicht und geben Sie auch Ihren Freunden keine Kopien. Damit stehlen Sie mir nämlich im wahrsten Sinne des Wortes das Brot aus dem Mund. So besteht die Gefahr, daß ich keine Möglichkeit mehr habe, einen Kompiler und andere Erweiterungen zu entwickeln. Wenn Sie ein registrierter Anwender sind, dann haben Sie unsere volle Unterstützung und Hochachtung.

So, das wär's. Danke fürs Lesen. Ich hoffe, daß Ihnen das Produkt gefällt und daß Sie viele schöne und kreative Stunden damit verbringen werden!

François



2: Jetzt gehts's los!

AMOS-Basic ist wirklich ein ganz erstaunliches Paket, und die Spiele, die Sie mit ihm schreiben können, übersteigen sogar Ihre kühnsten Träume. Alle Eigenschaften, die den Amiga so unwiderstehlich machen, sind in dieses fantastische System eingebaut worden. Mit der Hilfe von AMOS-Basic können Sie Spiele schreiben, die selbst die größten Profis in der Assembly-Programmierung ins Schwitzen brächten.

So können Sie zum Beispiel ganz problemlos bis zu 52 Hardware-Sprites gleichzeitig animieren. Das ist eine Glanzleistung, vor allem, wenn Sie bedenken, daß Ihnen die Amiga-Hardware selbst nur acht liefert.

Für noch mehr Action auf dem Bildschirm nutzen Sie einfach den Amiga Blitter-Chip. Blitter-Objekte können Sie in jedem beliebigen Grafik-Modus erstellen, einschließlich HAM! Die einzige Beschränkung für die Anzahl der Blitter-Objekte auf dem Bildschirm besteht durch den verfügbaren Speicher.

Jede beliebige Kombination der Amiga Grafik-Modi kann sofort auf dem Bildschirm dargestellt werden. Hardware-Scrolling ist nicht nur möglich, sondern ganz einfach! Durch einen eingebauten SCREEN OFFSET Befehl können Sie den ganzen Ablauf sofort ausführen.

Eigentlich besteht die einzige Schwierigkeit bei AMOS darin, einen Anfang zu finden. AMOS unterstützt mehr als 500 Basic-Befehle und wenn Sie bisher noch nie mit Basic gearbeitet haben, dann wird Sie diese Anzahl vielleicht etwas erschrecken. Und wenn Sie sich auf fremdes Gebiet wagen, dann ist es immer ganz gut, einen kundigen Führer dabei zu haben, der Ihnen ein Bißchen zeigt, wo's langgeht. Und eben das ist das Ziel dieses Kapitels.

Erst Sichern, dann Starten!

Bevor wir weitermachen, müssen Sie unbedingt alle AMOS Disketten kopieren. Dann können Sie mit dem System nach Herzenslust spielen, und riskieren nicht, dabei etwas Wichtiges zu löschen. Schlimmstenfalls können Sie bei EuroPress Software gegen eine geringe Gebühr Ersatzdisketten anfordern. Aber währenddessen können Sie natürlich nicht mit AMOS arbeiten.

Bei der Installation kommt es nun darauf an, welche Konfiguration Sie haben, aber länger als ein paar Minuten dauert es eigentlich nie.

Das Installieren von AMOS auf einem Einzellaufwerk

1. Formatieren Sie zwei leere Disketten für Ihre Kopie von AMOS-Basic.
2. Nehmen Sie die AMOS-Programmdiskette und schieben Sie den Schreibschutz nach rechts, dann ist ein kleines Loch zu sehen. Jetzt ist die Diskette während des Kopierens vor Fehlern geschützt.
3. Starten Sie den Amiga mit der normalen Workbench-Diskette.
4. Legen Sie die AMOS-Programmdiskette in das interne Laufwerk ein und wählen Sie das Icon mit der Maus an.

5 Aus dem Start-Menü wählen Sie jetzt das Kopieren von Disketten und folgen den Anweisungen auf dem Bildschirm. Bei Problemen schlagen Sie im Amiga- Handbuch nach. Dort finden Sie eine ausführliche Beschreibung.
6 Nun wiederholen Sie die Schritte 4 und 5 und kopieren Sie die AMOS Datendiskette.
7 Nun müssen Sie Ihre AMOS Originaldisketten nur noch sicher aufbewahren.

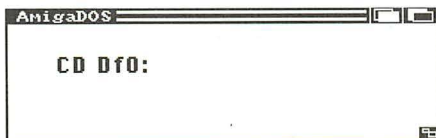
Das Installieren von AMOS auf einem Doppellaufwerk

1 Formatieren Sie zwei leere Disketten für Ihre Kopie von AMOS-Basic.
2 Nehmen Sie die AMOS-Programmdiskette und schieben Sie den Schreibschutz nach rechts, dann ist ein kleines Loch zu sehen. Jetzt ist die Diskette während des Kopierens vor Fehlern geschützt.
3 Starten Sie den Amiga mit der normalen Workbench-Diskette.
4 Legen Sie die AMOS-Programmdiskette in das interne Laufwerk ein und wählen Sie das entsprechende Icon mit der Maus an.
5 Legen Sie eine leere Diskette in das zweite Laufwerk.
6 Ziehen Sie die Amos Original-Diskette über die neue Diskette.
7 Dann folgen Sie den Prompts für das Erstellen der Sicherungskopie.
8 Jetzt wiederholen Sie die Schritte 4 bis 7 für die AMOS-Datendiskette.
9 Bewahren Sie die AMOS-Basic Original-Disketten sicher auf.

Das Installieren von AMOS auf der Festplatte

Die spezielle Routine zur Installation finden Sie auf der AMOS-Programmdiskette. Und so geht's:

1 Den Amiga wie üblich über die Festplatte starten.
2 In CLI oder SHELL gehen.
3 Die AMOS-Programmdiskette in das interne Laufwerk einlegen.
4 Das interne Laufwerk folgendermaßen einstellen:



5 Jetzt die folgende Zeile des CLI-Prompts eingeben:



6 AMOS wird jetzt in den Speicher geladen, und Sie folgen nun einfach den Prompts für die Installierung von AMOS auf der Festplatte.

Das Laden von AMOS-Basic

Es wird Sie nicht überraschen, daß AMOS-Basic auf viele verschiedene Arten angesprochen werden kann. Man kann zum Beispiel AMOS direkt aus dem Workbench-Menü laden, indem man das Icon mit der linken Maustaste anwählt. Wenn Sie AMOS auf diese Art laden, können Sie mit der Amiga- und der A-Taste auf der Tastatur zwischen AMOS und Workbench hin- und herschalten.

Da die Workbench jedoch viel kostbaren Speicherplatz verbraucht, den Sie für Ihre Basic-Programme besser nutzen könnten, werden die Profi-Nutzer es wahrscheinlich vorziehen, AMOS mit der normalen Start-Sequenz zu laden. So können Sie mit AMOS die besten Ergebnisse erzielen.

Jetzt wollen wir AMOS laden:

- Amiga ausschalten und ungefähr 10 Sekunden warten.
- Die Kopie der AMOS-Programmdiskette (Diskette 1) in das interne Laufwerk einlegen.
- Amiga jetzt einschalten, dann wird AMOS automatisch in den Speicher geladen.
- Eine Taste drücken, damit der Informationskasten verschwindet und so ins AMOS-System einsteigen.

Das AMOS Tutorial

Beim Einstieg in AMOS-Basic sehen Sie zuerst das Editier-Fenster. Es ist ganz einfach zu benutzen, und wenn Sie schon ein Bißchen Ahnung von Computern haben, dann bedarf es keiner weiteren Erklärung. Experimentieren Sie doch nach Herzenslust, der AMOS Editor ist recht intelligent, und Sie können kaum irgendwelche schwerwiegenden Fehler machen.

Jetzt haben Sie das AMOS Editier-Fenster gesehen, und wir schauen uns nun einige der Eigenschaften an, durch die AMOS wirklich aus der Masse hervorsticht.

Das Laden eines Programmes

Am Anfang wollen wir Ihnen zeigen, wie Sie eines der tollen Spiele von der AMOS Datendiskette laden können. Da haben wir zum Beispiel das "Number Leap"-Spiel:

- AMOS Datendiskette in das Laufwerk Df0: (das interne Laufwerk) einlegen.
- Amiga-Taste und gleichzeitig Taste "L" drücken. Dann erscheint die Datei-Liste zur Auswahl .
- In der Mitte der Datei-Liste befindet sich eine Reihe von Programmen, die in AMOS-Basic geladen werden können.
- Zur Auswahl des Number Leap Programms gehen Sie mit dem Maus-Zeiger auf die Datei:

Number_Leap.AMOS

Die gewählte Datei wird hervorgehoben.

- Jetzt können Sie die Datei durch zweimaliges Klicken mit der linken Maustaste in den Amiga Speicher laden, und der ursprüngliche Editor-Bildschirm erscheint wieder. Dieses Fenster enthält jetzt Ihr neues Programm-Listing.
- Wählen Sie "RUN" im Hauptmenü (oder wenn Sie faul sind, drücken Sie einfach F1).

Und jetzt geht's los: der Editor-Bildschirm verschwindet völlig und Number_Leap läuft vor Ihren Augen ab. Wenn Sie genug gespielt haben, drücken Sie gleichzeitig CTRL + C und steigen aus AMOS-Basic aus.

Mit Ctrl + C können Sie aus den meisten AMOS-Programmen aussteigen. Sollten Sie diese Abbruch-Möglichkeit aus irgendeinem Grund abschalten wollen, so geht dies mit dem BREAK OFF Befehl. Ist die Möglichkeit aber noch vorhanden, so kommt man mit einem Druck der Leertaste zurück in das Programm-Listing.

Das Löschen eines Programmes

Nachdem wir mit dem Number Leap Programm fertig sind, löschen wir es jetzt mit dem Befehl NEU aus dem Speicher. Sie finden diese Option nicht im Hauptmenü, sondern in einem getrennten SYSTEM-Menü. Wenn man mit dem Maus-Zeiger über das Menü-Fenster geht und dabei die rechte Maustaste drückt, dann erscheint das SYSTEM-Menü.

Programm löschen:

- Maus-Zeiger auf das Menü bringen.
- Rechte Maustaste drücken und SYSTEM-Menü aufrufen.
- Dann Taste gedrückt halten, auf NEU gehen und mit der linken Maustaste anwählen. Oder - für die Faulen - Shift + F9 drücken.
- Mit J bestätigen und N abbrechen (oder die Optionen mit der Maus anwählen).
- Wenn das gegenwärtige Programm nicht gespeichert wurde, erscheint die Frage, ob es auf der Diskette gespeichert werden soll. Wählen Sie JA, so erhalten Sie einen AMOS Datei-Selektor. Ansonsten wird das Programm total gelöscht.

Der Direktmodus

Jetzt schauen wir uns schnell den Direkt-Modus an. Das ist das Kernstück des AMOS Basic-Pakets, dadurch können Sie mit Ihren Routinen experimentieren und die Resultate sofort sehen. Der Knackpunkt ist, daß alle Schirme, Sprites und die Musik, die in Ihrem Programm definiert werden, völlig getrennt vom Editier-Fenster laufen. Deshalb ist es ganz egal, was sie im Direktmodus treiben, Sie können immer mit einem einzigen Tastendruck sofort wieder in Ihr Listing zurückgehen.

- In den Direktmodus gehen Sie mit Escape. Dann wird das Editier-Fenster durch die Anzeige des Hauptprogramms ersetzt.

Im unteren Teil dieses Bereichs ist ein kleiner Schirm, über den Sie Ihre Direct- Befehle eingeben können. Versuchen Sie es einmal mit der folgenden Zeile, und drücken Sie dann ENTER:

Print "Dein Name"

Wenn Sie Ihren Namen zwischen die Anführungsstriche setzen, dann erscheint er auf dem Bildschirm. Mit den Cursor-Tasten können Sie das Fenster auf dem Bildschirm bewegen. Wie Sie sehen, ist der Direktmodus vom Hauptschirm völlig unabhängig.

Animation!

Soviel zum Direktmodus. Nun wollen wir ein bißchen mit den Sprite-Anweisungen in AMOS-Basic experimentieren. Bevor wir diese Befehle jedoch anwenden können, müssen wir erst eine Reihe von Sprite-Bildern in den Speicher laden. Bleiben Sie dazu im *Direktmodus* und geben Sie nacheinander die eingerückten fettgedruckten Zeilen ein.

Auflisten der Sprite-Dateien

Zunächst listen wir alle verfügbaren Sprite-Dateien auf dem Bildschirm auf.

- Die AMOS Datendiskette muß im internen Laufwerk sein.
- Das Dateien-Verzeichnis folgendermaßen aufrufen:

Dir "AMOS_DATA:Sprites/"

So werden alle auf der AMOS Datendiskette befindlichen Sprite-Dateien aufgelistet. Diese Dateien enthalten alle Bilder, die in den verschiedenen Beispielprogrammen verwendet werden. Mit Hilfe des Sprite-Definierzusatzes (definer accessory) auf der AMOS-Programmdiskette können Sie auch Ihre eigenen Bilder erstellen.

Der Sprite-Definierzusatz umfaßt eine Vielzahl von leistungsstarken Design-Eigenschaften, mit denen es kinderleicht ist, für Ihre Spiele ganz profimäßige Animationssequenzen zu schaffen.

Laden einer Sprite-Datei

Und jetzt wollen wir diese Sprites laden. Sie gehen in eine spezielle Speicherbank, deshalb erwarten Sie bitte nicht, daß die Sprites sofort auf dem Bildschirm auftauchen! Mit dem folgenden Befehl laden wir die für das "Number Leap"-Spiel verwendeten Sprites:

Load "AMOS_DATA:Sprites/Frog_Sprites.abk"

Wenn Sie einen Fehler machen, dann gehen Sie mit F1 zur vorhergehenden Zeile zurück. Sie kann dann mit den normalen Cursor-Tasten korrigiert und ENTER nochmals eingegeben werden. Mit einem ähnlichen Befehl können wir auch eine Musik-Datei laden:

Load "AMOS_DATA:Music/Funkey.abk"

Wir prüfen dann mit dem Befehl LISTBANK nach, ob die Sprites und die Musik tatsächlich in den Speicher geladen wurden:

Listbank

Und es erscheint eine solche Liste:

1 - Sprites S:$0682B0 L:$000040
2 - Music S:$043878 L:$0081FE

Nur keine Angst, wenn die Zahlen in Ihrer Liste nicht so aussehen, sie hängen nämlich vom verfügbaren Speicher ab. Die Anzahl der Sprites, die wir gerade geladen haben, kann direkt mit der LENGTH-Funktion abgefragt werden.

Print Length(1)
64

In diesem Handbuch tauchen immer wieder Zeilen auf, die Sie eintippen können, und sie sind immer **fettgedruckt**. Der Text, der auf dem Computer erscheint, wird darunter in normaler Schrift dargestellt.

Das Einstellen der Sprite-Farben

Für jeden Satz von Sprite-Bildern ist ein Satz von Farbwerten auf der Diskette gespeichert. Da sich diese erheblich von Ihren gegenwärtigen Farben auf dem Bildschirm unterscheiden können, ist es nützlich, daß man die Farben aus der Sprite-Bank holen und in den Schirm kopieren kann. Dazu verwendet man den Befehl GET SPRITE PALETTE. Sie geben ein:

Get Sprite Palette

Alle Farben auf dem Hauptschirm verändern sich sofort, aber des Fenster für den Direktmodus bleibt unverändert, weil ihm vom AMOS System eine eigene Liste von Farbwerten zugewiesen wird.

Die Anzeige eines Sprites

Mit einem einfachen AMOS Basic-Befehl können Sprites überall auf dem Bildschirm dargestellt werden. Das geht zum Beispiel so:

Sprite 8,129,50,62

Animation eines Sprites

Und jetzt wollen wir das Sprite mit der AMOS Animations-Sprache (AMAL) animieren. AMAL ist ein einzigartiges Animations-System, mit dem Sie Ihre Objekte mit Wahnsinnsgeschwindigkeit bewegen oder animieren können. Bei der Eingabe der folgenden Beispielprogramme müssen Sie jede Zeile haargenau so eingeben, wie sie dargestellt ist, sonst könnten Sie unvermittelt einen Syntax-Fehler erhalten.

```
Sprite 8,129,50,62
Amal 8, "Anim 0,(62,5)(63,5)(64,5);" : Amal On
```

Mit diesem Programm wird die kleine Ente auf dem Bildschirm bewegt. Das Sprite kann währenddessen auch noch mit dem SPRITE-Befehl verschoben werden. Hier ist ein Beispiel:

```
Sprite 8,300,50,
```

Bewegen eines Sprites

Auf los geht's los!

```
Sprite 8,129,150,62 : A$="Anim 0,(62,5)(63,5)(64,5);"
A$=A$+"Loop: Move 320,0,100;"
Move -320,0,100; Jump Loop"
Amal 8,A$ : Amal On
```

Sie brauchen nur diese drei Zeilen, und schon wird die Ente lebendig und hüpft auf dem Bildschirm hin und her. Obwohl die Befehle in den Anführungszeichen wie Basic aussehen, werden sie doch in AMAL geschrieben. Alle AMAL Programme laufen fünfzigmal pro Sekunde ab, und so können Sie unabhängig von Ihrem Basic- Programm eine fließende Animation erreichen.

Um einmal zu sehen, wie atemberaubend AMAL wirklich ist, springen Sie doch mit Escape zurück in den Basic Editor. Nach ein paar Sekunden gehen Sie in den Direktmodus zurück, und da hüpft Ihre Ente immer noch herum, als sei nichts geschehen.

Musik, Herr Kapellmeister!

Und jetzt geht's mit Musik ins Finale! Sie sollten jetzt noch immer im Direktmodus sein. Jetzt drehen Sie den Monitor so laut wie möglich, dann machen Sie mit dem MUSIC-Befehl Musik:

```
Music 1
```

Und wenn es den Nachbarn zuviel wird, schalten Sie so wieder aus:

```
Music Off
```

Die Reise geht weiter

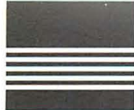
Hoffentlich haben Sie jetzt schon eine ganz gute Vorstellung davon, was AMOS- Basic so alles kann. Aber bis jetzt haben wir erst einen klitzekleinen Teil des mächtigen AMOS-Basic gesehen. Wenn Sie erst richtig mit AMOS experimentieren, dann entdecken Sie bald eine ganz neue Welt voll aufregender Möglichkeiten.

AMOS-Basic kann Sie natürlich nicht über Nacht zum Programmierexperten machen. Wie bei jeder anderen Programmiersprache dauert es eine Weile, bis man mit der gesamten Bandbreite der Befehle vertraut ist. Deshalb wollen wir Ihnen zum Abschluß dieses Kapitels noch ein paar Tips mit auf den Weg geben.

Tips und Tricks

- Am besten lernt man AMOS kennen, indem man kleine Programme schreibt, mit denen man Sprites animiert, Bildschirme scrollt oder Punktstandtabellen erstellt. Wenn Sie sich dann etwas mehr zutrauen, können Sie diese Routinen in ein Spiel einbauen.
- Lassen Sie sich von Umfang der AMOS Basic-Sprache keinen Schrecken einjagen. Sie können schon mit einem winzigen Teil der ca. 500 AMOS-Befehle Wahnsinnseffekte erzielen. Fangen Sie damit an, mit ein paar Befehlen wie SPRITE oder BOB sicher umzugehen, und arbeiten Sie sich dann langsam durch die verschiedenen Kapitel. So bauen Sie Ihr Wissen Schritt für Schritt auf.
- Obwohl wir uns bemüht haben, dieses Paket so benutzerfreundlich wie möglich zu gestalten, ist trotzdem eine gute Grundlage der allgemeinen Richtlinien der Programmierung in Basic unglaublich wertvoll. Wenn Sie noch keine Ahnung von Basic haben, dann hilft Ihnen ein Einführungstext wie z.B. **Alcock's Illustration Basic (Cambridge University Press)**weiter.
- Planen Sie Ihre Spiele erst sorgfältig auf dem Papier. Es ist ganz erstaunlich, wieviele Probleme so schon im Anfangsstadium vermieden werden können. Versuchen Sie nie, größere Projekte ohne Vorbereitung in Angriff zu nehmen, denn das ist der kürzeste Weg in die Wildnis.
- Wenn Sie ein Spiel schreiben, konzentrieren Sie sich in erster Linie auf das Spielen und weniger auf die Spezialeffekte. Wenn die Idee erst einmal ausgereift ist, können Musik und Grafik nachher immer noch hinzugefügt werden.
- Füllen Sie Ihre Registrierkarte aus und treten Sie sofort dem AMOS-Club bei. Dann erhalten Sie regelmäßig unser Informationsblatt mit neuen Ideen, Tips und Tricks und erfahren das Neueste über AMOS-Basic. Außerdem können Sie auf eine reiche Auswahl von Public Domain Software, einschließlich Sprites, Kostproben von Spielen und Musik zugreifen, die Sie alle in Ihre eigenen Programme einbauen können.

Es wird sich in der Zukunft viel in der AMOS-Szene tun, so wird es zum Beispiel bald Zusatzprogramme, Utilities und auch Bücher geben. Wenn Sie also auch eine wichtige Rolle bei der Weiterentwicklung dieses Programms spielen wollen, dann treten Sie **jetzt** gleich dem AMOS-Club bei! Wir freuen uns darauf.



3: Der Editor

Der AMOS-Editor bietet Ihnen eine riesige Auswahl an Editier-Möglichkeiten. Er ist nicht nur äußerst leistungsstark, sondern es macht richtig Spaß, mit ihm zu arbeiten. Alle Befehle werden entweder direkt auf dem Bildschirm oder über eine eindrucksvolle Reihe von Alternativen über die Tastatur eingegeben. Der Editor ist so bedienerfreundlich, daß Sie ihn wahrscheinlich - selbst wenn Sie noch nicht soviel Erfahrung im Umgang mit Computern haben - sofort einsetzen können.

Eine der tollsten Eigenschaften dieses Systems ist die Darstellung der Programm-Liste vollkommen getrennt vom Hauptschirm. Somit können Sie durch einen einzigen Tastendruck (ESCape) ganz einfach zwischen Ihrer Programmanzeige und dem Editier-Fenster hin- und herschalten.

Wenn Sie viel Speicherplatz haben, können Sie auch mehrere AMOS Basic-Programme gleichzeitig laden. Jedes Programm kann völlig unabhängig editiert werden, und man kann problemlos zwischen den verschiedenen Programmen im Speicher durch Drücken von zwei Tasten im Editor hin- und herspringen.

Wenn AMOS in den Speicher geladen wird, so erscheinen zuerst die Namen aller, die an der Entwicklung teilgenommen haben. Applaus, Applaus! Jetzt eine Taste drücken und in den Editor gehen.

Das Menü-Fenster

Oben auf dem Bildschirm sehen Sie ein Menü-Fenster mit den verfügbaren Optionen. Es ist das Tor zum Reich der mächtigen Editier-Funktionen von AMOS- Basic. Die Befehl werden immer sofort ausgeführt. Sie wählen sie entweder mit der linken Maus- oder einer entsprechenden Funktions-Taste an.

Zusätzlich zum Hauptmenü gibt es natürlich eine Reihe anderer Menüs. Das wichtigste unter ihnen ist das SYSTEM-Menü. Das ruft man entweder durch Drücken der rechten Maustaste oder der Shift- (Umschalt-)Taste auf.

Das SYSTEM-Menü enthält viele unentbehrliche Befehle wie LADEN, SICHERN, NEU usw. Wie im Hauptmenü können alle Funktionen entweder mit der linken Maustaste oder der entsprechenden Funktions-Taste gewählt werden. Hier sind einige Beispiele:

F1	Aktuelles Programm ablaufen lassen
F2	Programm auf Syntax-Fehler überprüfen
Shift+F1	Programm laden (auch Amiga+L)
Shift+F3	Programm speichern (auch Amiga+S)

Unmittelbar unter dem Menü-Fenster steht eine Zeile mit wichtigen Informationen.

Die Status-Zeile

I **L:**=1 **C**=1 **Text**=40000 **Chip**=91000 **Fast**=0 **EDIT**=Beispiel

Die Anzeige ganz links ist für den Editier-Modus (Insert = Einfügen oder **O**verwrite = Überschreiben). Daneben werden die Zeile (**L**ine) und die Spalte (**C**olumn), die Sie zur Zeit bearbeiten, angezeigt. Die anderen Meldungen bedeuten:

TEXT: Zeigt den Speicherplatz an, der dem Editier-Fenster zugewiesen wurde. Man kann ihn über den einfachen AMOS Basic-Befehl TEXTPUFFER aus dem SUCH- Menü einstellen.

CHIP: Das ist der Speicher, auf den die Amiga-Chips zugreifen können. Erschrecken Sie nicht, wenn Sie einen A500 ohne Speichererweiterung haben und meinen, Sie haben viel zu wenig Speicherplatz frei. Es gibt da ein paar Möglichkeiten, mit denen Sie diesen Wert in Ihren Basic-Programmen drastisch erhöhen können (Einzelheiten erfahren Sie im Abschnitt über SPEICHERVERWALTUNG).

FAST: Das ist der in Ihrem Computer vorhandene FAST-Speicher. Wenn möglich, adressiert AMOS immer diesen Speicher anstatt des wertvollen Chip-Speichers.

EDIT: Hier wird das Programm, das Sie gerade editieren, angezeigt. Dieses Feld bleibt leer, bis Sie ein Programm laden oder speichern, dann zeigt es den entsprechenden Datei-Namen automatisch an.

Das Editier-Fenster

Das Editier-Fenster stellt den Kern des AMOS-Systems dar, und ermöglicht Ihnen die direkte Eingabe in Ihre Basic Programm-Listings über die Tastatur. Der gesamte Text wird an der gegenwärtigen **Cursor-Position** eingefügt, man erkennt sie an der blinkenden, horizontalen Linie.

Zu Beginn der Arbeit steht der Cursor immer in der linken oberen Ecke des Editier-Fensters. Er kann in der aktuellen Zeile mit den Cursor-Tasten nach rechts und links bewegt werden.

Die Zeile können Sie jetzt Zeichen für Zeichen mit der Lösch- (Del) und der Rücktaste (Backspace) korrigieren. Die Löschtaste löscht das Zeichen unmittelbar unter dem Cursor, die Rücktaste das Zeichen links vom Cursor. Tippen Sie zum Beispiel:

Print "AMOS"

Wenn Sie jetzt ENTER drücken, dann wird diese Zeile in AMOS-Basic eingegeben. Alles, was AMOS als Befehl erkennt, wird sofort in ein spezielles Format übertragen. Bei allen Basic-Befehlen wird der Anfangsbuchstabe groß, und die restlichen Buchstaben klein geschrieben. Die o.g. Zeile wird also folgendermaßen angezeigt:

Print "AMOS"

Alle AMOS Variablen und Prozeduren werden in GROSSBUCHSTABEN dargestellt. So

können Sie schnell feststellen, ob Sie in einer Programmzeile einen Fehler gemacht haben. Nehmen wir einmal an, Sie haben folgende Zeile eingegeben:

```
Inpit "Wie heißen Sie;";name$
```

Das wird dann so dargestellt:

```
INPIT "Wie heißen Sie;";NAME$
```

Da INPIT GROSS geschrieben wird, ist offensichtlich, daß Sie einen Fehler gemacht haben.

Und jetzt wollen wir ein bißchen spielen. Gehen Sie mit dem Cursor unter den Print-Befehl, den Sie vor ein paar Minuten eingegeben haben, und tippen Sie die folgenden Basic-Anweisungen ein:

```
Centre "<Klick und Tipp Demo>"
 Do
    X$=Inkey$ : If X$<>"" Then Print X$;
Loop
```

Vergessen Sie nicht, nach jeder Zeile ENTER zu drücken. Jetzt gehen Sie Ihr neues Programm mit den Cursor-Tasten durch. Zum Schluß drücken Sie F1, und Ihr Programm läuft ab.

Das Editier-Fenster verschwindet, und eine eigene Programm-Anzeige taucht stattdessen auf. Jetzt erwartet das Programm, daß Sie über die Tastatur Text eingeben. Wie Sie sehen, hat der Programm-Schirm seine eigene Cursor-Zeile. Diese ist von der Cursor-Zeile im Editor ganz unabhängig, und Sie können jetzt herumspielen, soviel Sie wollen, ohne Ihre Position im Editier-Fenster zu verändern. Wenn Sie fertig sind, brechen Sie das Programm mit Ctrl+C ab. Dann erscheint eine dünne Linie auf dem Bildschirm, die mit den Cursor-Tasten herauf- und herunterbewegt werden kann.

```
        Programm unterbrochen in Zeile 4
>>>Loop
```

Mit der Leertaste könnten Sie jetzt wieder in den Editor zurückgehen. Aber den kennen wir ja nun schon, deshalb wollen wir uns lieber schnell den Direktmodus ansehen. Dazu drücken wir Escape.

Einführung in den Direktmodus

Der Direktmodus hilft Ihnen auf einfache Art, Ihre Basic-Programme zu testen. Vorerst wollen wir uns nur einige seiner interessanten Eigenschaften ansehen.

Alle Befehle in diesem Modus werden über einen speziellen Schirm, der von der Programm-Anzeige völlig unabhängig ist, eingegeben. Mit den Cursor-Tasten können

Sie diesen Schirm nach oben oder unten schieben. Oben im Fenster finden Sie eine Liste von 20 Tastenfunktionen. Das sind die Befehle, die den Funktionstasten zugeordnet wurden. Zum Aufrufen drückt man eine der Amiga- Tasten zusammen mit der entsprechenden Funktionstaste.

Im Direktmodus können Sie jede beliebige Basic-Anweisung ausführen, ausgenommen Schleifen (Loops) oder Prozeduren (procedures). Wie im Editor wird auch hier die Eingabe der Befehle mit der ENTER-Taste abgeschlossen. Hier folgen einige Beispiele:

Print 42	Zeigt eine Konstante an
ANTWORT=6: Print ANTWORT*9	Berechnung durchführen
Curs Off	Cursor abschalten
Close Workbench	Workbench abschalten. Spart etwa 40KB,bricht aber Multitasking ab.
Run	Programm nochmal ablaufen lassen.

Vergessen Sie nicht, ganz gleich, was Sie im Direktmodus anstellen, es hat absolut keine Auswirkungen auf das aktuelle Programm-Listing. Deshalb können Sie nach Belieben herummurksen, und es besteht keine Gefahr, daß Sie irgend etwas in Ihrem Basic-Programm löschen.

Jetzt wird es aber Zeit, daß wir ins Editier-Fenster zurückgehen. Verabschieden wir uns also und drücken Escape.

Und, was habe ich gesagt? Der Cursor blinkt noch an derselben Stelle wie vorher. Sie sehen, die zwei Modi sind wirklich total unabhängig voneinander. Zum Test drücken Sie nochmal Escape und schwupps, schon sind Sie wieder im Direktmodus. Wenn Sie dann vom Direktmodus genug haben, gehen Sie mit Escape wieder ins Editier-Fenster.

Das Laden eines Programms

Jetzt wollen wir die verschiedenen Möglichkeiten zum Laden und Speichern Ihrer Programme auf einer Diskette besprechen. Wie üblich, können die Optionen entweder über das Menü-Fenster oder einfache Tastenkombinationen im Editor ausgeführt werden. Am schnellsten lädt man ein Programm, wenn man eine der Amiga-Tasten und gleichzeitig L für Laden drückt.

Dann sehen Sie das Fenster für den AMOS Datei-Selektor. Heutzutage sind diese Fenster in den meisten für den Amiga verfügbaren Software-Paketen enthalten. Wenn Sie schon einmal mit einem solchen Fenster gearbeitet haben, dann werden Sie auch mit dem AMOS-System keine Überraschungen erleben. Aber wir wollen dieses Fenster doch noch etwas genauer erklären.

Der AMOS Datei-Selektor

Die Auswahl einer Datei von der Diskette ist wirklich kinderleicht. Einfach mit dem Maus-Cursor den gewünschten Dateinamen ansteuern, so daß er hervorgehoben wird, und dann zweimal mit der linken Maustaste anklicken. Oder den Dateinamen über die Tastatur eingeben und ENTER drücken.

Wenn Sie einen Fehler machen, oder aus dem Datei-Selektor aussteigen möchten,

ohne eine Datei zu laden, gehen Sie mit der Maus auf "Ende" und klicken diese Option mit der linken Maustaste an. Dann bricht AMOS diese Operation ab, und in der Status-Zeile wird "Nicht erledigt" angezeigt.

Legen Sie doch jetzt einmal Ihre KOPIE der AMOS-Programmdiskette in das interne Laufwerk und drücken Sie Amiga+L zum Laden einer Datei. Wenn Sie bisher mit unserem Tutorial gearbeitet haben, wird AMOS Ihnen zuerst die Möglichkeit geben, Ihr aktuelles Programm zu speichern. Sofern Sie keine interessanten Veränderungen vorgenommen haben, drücken Sie "N", und gehen in den Datei-Selektor. Ansonsten schlagen Sie am besten unter Das Speichern eines Programms nach.

Wenn der Datei-Selektor auftaucht, dann suchen Sie eine Datei mit dem Namen "Hallo.AMOS" heißt. Wenn Sie sie gefunden haben, dann gehen Sie mit der Maus auf den Namen und klicken ihn zweimal mit der linken Maustaste an.

Dann wird die Beispiel-Datei in den Speicher geladen und das folgende Listing erscheint in AMOS-Basic:

```
Rem Hallo AMOS Anwender !
Cls 0 : Rem Löschen des Bildschirms mit Farbe 0
Do
    Rem Ein paar Zufallszahlen ausdenken
    X=Rnd(320):Y=Rnd(200):I=Rnd(15):P=Rnd(15)
    Ink I,P : Rem Und ein bißchen Farbe hinzu ...
    Text X,Y"Hallo !" : Rem ... und fertig !
Loop
```

Gehen Sie mit dem Cursor auf "Hallo !" und fügen Sie Ihre eigene Begrüßung ein. Dann drücken Sie F1 und lassen das Programm ablaufen. Auf der Programmanzeige wird Ihr Text im Nu x-mal kopiert. Mit Ctrl+C können Sie aus dieser Routine wieder aussteigen.

Das Speichern eines Basic-Programms

Gehen Sie ins Editier-Fenster zurück und geben Sie Shift+F3 ein, um Ihr Programm auf Diskette zu speichern. Oder wenn Ihnen mal nach etwas anderem zumute ist, dann halten Sie die rechte Maustaste gedrückt und klicken die Option Save As des SYSTEM-Menüs mit dem linken Maus-Knopf an. So oder so, auf alle Fälle werden Sie wieder im AMOS Datei-Selektor-Fenster landen.

Jetzt geben Sie am besten den Namen Ihrer neuen Datei über die Tastatur ein, dann erscheint er in einem kleinen Fenster unten im Datei-Selektor. Wie beim Editor steht auch hier der Cursor an der Eingabeposition, und er kann mit den üblichen Tasten bewegt werden. Zu guter Letzt drücken Sie dann ENTER, um Ihr Programm zu speichern. Das war doch wirklich kinderleicht, oder?

Durchsehen der Dateiliste

Wenn Ihre Diskette verhältnismäßig voll ist, dann passen die Dateien nicht alle gleichzeitig ins Selektor-Fenster. Mit dem Scroll-Balken links am Selektor-Fenster können Sie die Liste durchgehen.

Dazu die Maus auf dem Balken plazieren und die linke Taste gedrückt halten. Jetzt

können Sie den Balken mit der Maus 'rauf- und 'runterziehen und das Selektor- Fenster so bewegen. Eine ähnliche Wirkung hat das Anklicken des entsprechenden Pfeil-Icons.

Ändern des aktuellen Laufwerks

Klicken Sie die rechte Maustaste an, wenn ein Datei-Requester angezeigt wird, so ersetzt eine Liste der Device-Namen das aktuelle Dateiverzeichnis. Sie erhalten dann einfach durch Anklicken des gewünschten Devices Zugang zu ihm. Dann führt AMOS alle über dieses Device verfügbaren Dateien auf. Der Inhalt des Fensters hängt natürlich davon ab, über welche Ausrüstung Ihr Amiga verfügt.

Ändern des Verzeichnisses

Wenn Sie Ihre Dateiliste durchsehen, dann entdecken Sie, daß einige der Dateien mit einem Sternchen * markiert sind. Das sind aber gar keine Dateien, sondern eigene Verzeichnisse (directories).

In so ein Unterverzeichnis kommen Sie auch wieder mit der linken Maustaste, und dann können Sie sich die Dateien in diesem Verzeichnis ansehen. Dabei ist zu beachten, daß nur die Dateien mit der aktuellen Erweiterung ".AMOS" aufgeführt werden.

Wenn Sie ein Verzeichnis auf diese Art geöffnet haben, können Sie es auch mit der V.Ändern-Taste als Default konfigurieren. Wenn Sie dann das nächste Mal den Datei-Selektor aufrufen oder mit DIR die Liste anzeigen lassen, erscheint automatisch zuerst der Inhalt des gewählten Unterverzeichnisses. Über die PARENT-Taste können Sie natürlich auch wieder in das Hauptverzeichnis zurückgelangen. Hier handelt es sich um eine kleine, runde Icon-Taste direkt über dem Pfeil nach oben. Weitere Einzelheiten finden Sie unter dem PARENT-Befehl.

Das Sortieren des Verzeichnisses

So wie AMOS das Verzeichnis über ein Device einliest, ermöglicht AMOS es Ihnen auch, jederzeit eine Datei auszuwählen. Diese Methode ist wesentlich schneller, hat jede die Nebenwirkung, daß das Verzeichnis nicht sortiert wird. Deshalb weist AMOS im Datei-Requester die leistungsstarke SORTIERE-Taste auf, und so können Sie dann schnell die gewünschte Datei finden.

Setzen des Suchpfades

Normalerweise sucht AMOS nach allen Dateinamen mit der Erweiterung ".AMOS". Wenn Sie eine Datei mit einer anderen Erweiterung laden wollen, z.B. .BAK, dann machen Sie das folgendermaßen:

Gehen Sie mit dem Cursor in das PFAD-Fenster, indem Sie mit der Cursor-Taste nach oben gehen. Jetzt tippen Sie Ihren neuen Pfad ein und drücken ENTER. Ein genaue Beschreibung der Befehls-Syntax finden Sie in dem Abschnitt zum DIR- Befehl.

Achtung! AMOS verwendet seine eigenen Suchkriterien, die vom normalen Amiga-Dos-System stark abweichen. Wenn Sie sich nicht sicher sind, dann löschen Sie lieber

alles bis zur aktuellen VOLUME- oder DRIVE-Bezeichnung und drücken Sie ENTER. Dann erhalten Sie das gesamte Verzeichnis aller Dateien.

Anwenden des Datei-Selektors

Ganz besonders interessant dabei ist, daß Sie den Datei-Selektor direkt aus Ihren eigenen Programmen aufrufen können. Machen wir doch eine kleine Probe aufs Exempel: Sie gehen in den Direktmodus (Escape drücken) und geben die folgende Zeile ein:

```
Print Fsel$(*.*)
```

Nachdem Sie eine Datei gewählt haben, wird der Name, den Sie ausgesucht haben, auf dem Bildschirm erscheinen. Unter FSEL$ finden Sie Einzelheiten zu diesem Befehl.

Das Tutorial zum Editor

Jetzt wollen wir einen kurzen Blick auf die Editier-Möglichkeiten für Fortgeschrittene werfen. Zuerst laden wir dazu ein Beispielprogramm von der Diskette. Damit's ein wenig spannender wird, haben wir sie in ein anderes Unterverzeichnis namens MANUAL auf der AMOS-Programmdiskette gepackt.

Also, legen Sie Ihre KOPIE!! der AMOS-Programmdiskette ins interne Laufwerk ein und rufen Sie den Datei-Selektor mit Amiga+L auf. Dann öffnen Sie das MANUAL-Unterverzeichnis durch Anwählen mit der linken Maustaste. Und wie Sie sehen, gibt es für jedes Kapitel in diesem Handbuch ein eigenes Unterverzeichnis.

Die Dateien zu diesem Kapitel finden Sie im Unterverzeichnis "Kapitel_3". Einfach den Cursor auf dieses Feld stellen, dann mit der linken Maustaste anklicken. Sie erhalten nun eine Aufstellung aller Beispiel-Dateien für dieses Kapitel. Na, dann wollen wir mal: laden Sie die Datei **Beispiel 3.1.AMOS** in den Speicher.

Für die ganz Neugierigen: das ist ein kleines Programm, das eine funktionstüchtige Dialog-Box auf dem Bildschirm darstellt. Drücken Sie F1 für eine schnelle Demo. Wenn Sie eine der Tasten anklicken, dann zeigt das Programm nur ihre Nummer an und geht dann in AMOS-Basic zurück.

Wenn Sie das Programm genauer unter die Lupe nehmen, dann stoßen Sie auf ein paar wichtige Punkte. Erstens, es ist nirgends eine Zeilenangabe zu sehen. AMOS-Basic ist so stark, daß Zeilenangaben einfach überflüssig sind. Deshalb lassen wir Sie entscheiden, ob Sie sie möchten oder nicht. Eine andere interessante Eigenschaft besteht darin, daß viele Zeilen mit einer eigenartigen "Procedure"- Anweisung beginnen. Sie stellen die Kopfzeilen für die Prozedur-Definitionen in allen AMOS- Programmen dar.

Prozeduren sind unabhängige Programmteile mit einer eigenen Liste von Variablen und Datenangaben. Sie sind eigentlich den GOSUBs im Standard-Basic sehr ähnlich. Aber sie können viel, viel mehr! In Kapitel 4 werden wir sie uns noch näher ansehen.

Es gibt verschiedene Möglichkeiten, wie Sie das Programm nun genauer betrachten können:

Das Scrollen durch ein Listing

Neben dem Hauptfenster des Editors sind zwei "Scroll-Balken" angeordnet. Mit der Maus können Sie über diese Balken in Ihrem Listing blättern. Dazu gehen Sie mit der Maus auf den vertikalen Balken und halten die linke Taste gedrückt. Jetzt ziehen Sie den Balken auf dem Bildschirm nach unten und das Editier-Fenster rollt glatt nach unten durch das Listing. Sie können aber auch mit dem entsprechenden Pfeil-Icon oben bzw. unten am Balken im Listing blättern. Das Anklicken eines Icons schiebt die Zeile genau einen Schritt in die entsprechende Richtung. Am unteren Ende des Editier-Fensters ist ein horizontaler Balken angebracht. Mit ihm kann man das Fenster auf dieselbe Art nach rechts und links bewegen.

Und wenn Sie lieber tippen, dann können Sie das Ganze auch folgendermaßen über die Tastatur durchführen:

• Pfeil nach oben (Tastatur)	Fenster um eine Zeile nach oben schieben
• Pfeil nach unten	Fenster um eine Zeile nach unten schieben
• Ctrl + Pfeil nach oben	Verschiebt das Listing auf die vorherige Seite
• Ctrl + Pfeil nach unten	Bewegt das Listing auf die nächste Seite

Alle Tastatur-Optionen unterliegen demselben Grundprinzip. Also, wenn Sie einmal einen Befehl beherrschen, dann ist der Rest ganz einfach. Am Ende dieses Kapitels finden Sie eine Aufstellung aller Befehle.

Und jetzt haben wir genug gesehen, nun sollen Taten folgen, also suchen Sie in Ihrem Programm-Listing nach der Zeile:

```
ALERT[50,"Alert box","","Ok","Cancel",1,2]
```

Damit wird eine Basic-Prozedur aufgerufen, die eine funktionierende Alarm-Box auf den Bildschirm bringt. Das Format dieser Prozedur lautet:

```
ALERT[Y coord,Titel1$,Titel2$,Knopf1$,Knopf$,Papier,Tinte]
```

Und jetzt wollen wir diese Alarm-Box etwas aufpeppen. Gehen Sie mit dem Cursor über die obengenannte Anweisung und korrigieren Sie die Zeile mit den Cursor- Tasten, bis sie so aussieht:

```
ALERT[50,"Erledige!","Stephen","Los!","Nicht doch!",13]
```

Lassen Sie das Programm ablaufen - dazu entweder F1 drücken oder aus dem Hauptmenü RUN wählen. Und dann haben Sie die einzigartige Gelegenheit, den Verfasser dieses Handbuchs lahmzulegen. Taste anwählen und los gehts. Autsch, das hat gesessen!

In der Praxis können Sie die Bezeichnungen und Tasten beliebig verändern. Vielleicht möchten Sie diese kleine Routine auch in eines Ihrer eigenen Programme einbauen.

Hoffentlich wollen Sie jetzt selbst Prozeduren in Ihren eigenen Programmen einsetzen, denn dazu wollten wir Sie eigentlich mit diesem Kapitel motivieren. Und um Ihnen dabei unter die Arme zu greifen, haben wir in den AMOS Editor unwahrscheinlich starke Spezialfunktionen eingebaut.

Das Suchen nach Sprungmarken und Prozeduren

Wenn Sie ein sehr langes Programm haben, dann ist es oft nicht einfach, den Anfang der verschiedenen Prozedur-Definitionen zu finden. Deshalb haben wir die Möglichkeit eingebaut, direkt mit Alt+Cursor-Taste zur nächsten Definition in Ihrem Programm zu springen.

Probieren Sie es doch aus: Sie gehen zum Anfang der Liste und drücken Alt+Pfeil nach unten. Dann springt der Cursor sofort zum Anfang der ersten Prozedur- Definition dieses Programms (ALERT). So können Sie bis zum Ende der Liste springen und mit Alt+Pfeil nach oben wieder hinauf zum Anfang.

Dieses System ist natürlich nicht auf Prozeduren beschränkt, es funktioniert genauso gut bei Sprungmarken (Labels) oder Zeilennummern. Also können Sie damit auch arbeiten, wenn Sie mit Prozeduren nichts am Hut haben.

Das Falten von Prozedur-Definitionen

Wenn Sie Ihre Programme aus einer Liste häufig verwendeter Prozeduren zusammenstellen, können Ihre Listings mit den Definitionen Ihrer verschiedenen Library-Routinen vollgestopft werden.

Aber hier ist glücklicherweise Hilfe in Sicht. Wir rufen einfach den FALTEN-Befehl auf und Sie können Ihre Prozedur-Definitionen verbergen. Die Routinen können wie sonst auch in Ihrem Programm eingesetzt werden, aber ihre Definitionen werden durch eine einzige Prozedur-Anweisung ersetzt. Dazu folgendes Beispiel:

Gehen Sie mit dem Cursor an eine beliebige Stelle in der Definition von ALERT und klicken Sie den FALTEN-Befehl im Menü-Fenster an. Schwupps! Schon hat sich der Inhalt Ihrer Prozedur in Luft aufgelöst. Und dieses Mittel ist völlig ohne Nebenwirkungen auf das Programm. Die einzige Veränderung ist in dem Listing im Editier-Fenster zu bemerken.

Wenn Sie diese Prozedur überarbeiten möchten, dann wählen Sie einfach FALTEN nochmal an, und schon erscheint die gesamte Prozedur wieder in voller Schönheit.

Sie können aber auch ALLE Prozeduren Ihres Programms auf einen einzigen Streich verbergen. Das geht durch die Option *Alle öffnen* im SUCH-Menü. Das SUCH-Menü rufen Sie durch Anklicken mit der Maus oder über die Funktionstaste F5 auf. Mit *Alle zu* können Sie jetzt alle Prozedur-Definitionen aus dem geladenen Programm verschwinden lassen.

Probieren Sie es doch bei **BEISPIEL 3.1** aus, die Wirkung ist erstaunlich! Jetzt paßt das ganze Programm auf einen einzigen Schirm. Und Sie können auf einen Blick sehen, welche Prozeduren wir für das Programm verwendet haben. Jede der Prozedur-Definitionen kann einzeln über die Option FALTEN aufgerufen und dann überarbeitet werden, oder Sie wählen einfach *Alle öffnen* im SUCH-Menü und das ganze Programm wird wieder angezeigt.

Suchen/Ersetzen

Die Befehle Suchen/Ersetzen im AMOS-Basic Editor werden über ein spezielles SUCH-Menü aufgerufen, entweder im Menü-Fenster oder durch Drücken von F4.

Wer suchet, der findet..

Einige dieser Befehle sehen wir uns nun etwas genauer an. Wir wollen mit dem Befehl SUCHEN beginnen. Man kann ihn entweder über das SUCH-Menü oder Ctrl+F aufrufen. Dann müssen Sie die Suchfolge eingeben, und das geht z.B. so: Erst drücken Sie Ctrl+F, dann tippen Sie Rem.

Jetzt sucht AMOS nach der nächsten Bemerkung = Rem in Ihrem Programm, angefangen von der aktuellen Cursor-Position. Wenn AMOS dann fündig wird, springt der Cursor zum angegebenen Wort. Jetzt kann die Suche mit der Option *Weiter S.* (= Weiter Suchen) (Ctrl+N) wiederholt werden.

Ersetzen

Nehmen wir einmal an, wir möchten alle Rem-Anweisungen durch die entsprechenden "`"-Zeichen ersetzen. Dazu müssen wir nach Aufrufen von "Ersetzen" (Ctrl+R) die zu ersetzende Zeichenfolge eingeben. Wir drücken also Ctrl+R, tippen ` ein, und drücken ENTER. Und jetzt müssen wir erst das finden, was wir ersetzen wollen:

- Ctrl+F für Suchen drücken
- Rem eintippen
- Dann springt der Cursor zur nächsten Rem-Anweisung in Ihrem
- Programm.

Mit Ctrl+R (Ersetzen) wird sie jetzt ersetzt, und der Cursor springt zur nächsten Rem-Anweisung. Wenn die Rem-Anweisung jedoch in der Mitte der Zeile ist, dann müssen Sie unverrichteter Dinge weiterspringen, weil man in AMOS mit diesem Befehl nur etwas ersetzen kann, das am Anfang der Zeile steht. Dieses Problem kann man umgehen und stattdessen mit *Weiter S.* weiterspringen.

Ausschneiden und Einsetzen

Mit den Block-Befehlen können Sie Teile Ihres Programms ausschneiden und für den späteren Einsatz speichern. Wenn Sie einen Block einmal definiert haben, dann können Sie ihn im aktuellen Listing an jede beliebige Stelle kopieren.

Schauen wir uns das an einem Beispiel an: Wir nehmen das vorhergehende ALERT- Programm und schneiden eine Prozedur aus. Gehen Sie mit der Maus über die erste Zeile der INVERT-Prozedur und drücken Sie die rechte Maustaste.

Halten Sie sie gedrückt und ziehen Sie die Maus nach unten, dabei wird dann der gewählte Bereich hervorgehoben.
Mit dem Bl. Schneide-Befehl (BL=Block) (Ctrl+C) packen wir diesen Bereich jetzt in den Speicher. Und mit dem Bl. Einfügen-Befehl (Ctrl+P) kann dieser Block an einer beliebigen Stelle wieder ins Programm eingesetzt werden. In unserem Beispiel gehen wir jetzt mit dem Cursor an das Ende des Listings und rufen die Option *Bl. Einfügen* mit

Ctrl+P auf. Jetzt wird die INVERT-Prozedur dorthin kopiert.

Mehrere Programme und Zusätze im Speicher

Anzahl der Programme

Obwohl Sie mit AMOS immer nur jeweils ein Programm bearbeiten können, liegt die einzige Beschränkung für die Anzahl der im Speicher installierten Programme bei dem verfügbaren Speicherplatz. Wenn Sie ein Programm einmal im Speicher installiert haben, können Sie es gleich aus dem Editier-Fenster über die Option St. Andere (Starte Andere) ablaufen lassen.

Wenn Sie eine Speichererweiterungskarte eingebaut haben, dann können Sie leicht zwei oder drei Programme normaler Größe im Speicher haben. Aber auch wenn Sie nur einen A500 mit 512 KB Speicher haben, werden Sie diese Funktion gut nutzen können. Nehmen wir zum Beispiel einmal an, Sie stoßen in einem Ihrer Programme auf ein Problem. Dann können Sie Ihr derzeitiges Programm ganz einfach in den Speicher stecken und mit den verschiedenen Möglichkeiten herumexperimentieren, bis Sie eine Lösung gefunden haben. Wenn Sie fertig sind, dann packen Sie Ihre neue Routine mit der Bl. Scheide-Option in den Speicher und drücken nur zwei Tasten, schon sind Sie wieder in Ihrem Programm.

Dann können Sie die neue Routine einsetzen und schon gehts weiter. Sie werden sehen, die Möglichkeit, alles stehen- und liegenzulassen, und sofort eine Idee auszuprobieren, ist einfach unbezahlbar.

Außerdem ist es möglich, die am häufigsten gebrauchten Utilities wie den Sprite-Definer oder Map-Editor in den Speicher zu laden und somit immer zugriffsbereit zu haben.

Aber es geht sogar noch leichter, denn AMOS umfaßt ein ACCESSORY-System. So können die Utilities auf alle Speicherbanken in Ihren Hauptprogrammen uneingeschränkten Zugriff erhalten. Und der Sprite-Definer kann die Bilder direkt aus Ihrem aktuellen Programm holen und sie sofort bearbeiten. Mit dieser Technik wird der ganze Entwicklungsprozeß ungemein beschleunigt.

Das wollen wir uns doch auch in der Praxis ansehen. Geben Sie das folgende kleine Programm in den Editor ein:

```
Print "Das ist Programm Eins"
Boom
```

Mit dem Befehl *Push* schieben wir dieses Programm jetzt in den Speicher. Dazu drücken wir Amiga+P. Dann müssen Sie den Namen Ihres Programms in die Status-Zeile eingeben. Also tippen Sie hier zum Beispiel "Programm1".

Jetzt wird der Schirm völlig gelöscht, und das neue Fenster ist von Ihrem ursprünglichen Programm total unabhängig. Das sehen Sie, wenn Sie eine zweite Routine eingeben, z.B.:

```
Print "Das ist Programm Zwei"
Shoot
```

Dieses Programm können Sie jetzt im Editier-Fenster mit START (F1) ablaufen lassen. Aber Sie können dann auch sofort mit der Flick-Option in das vorherige zurückspringen. Jetzt drücken Sie Amiga+F und wie zuvor müssen Sie einen Programm-Namen eingeben. Wählen zum Beispiel Sie "Programm2". Und eins, zwei, drei, Zauberei: der Editor springt wieder in Ihr ursprüngliches Programm zurück.

Sie können jetzt den ganzen Ablauf wiederholen und zwischen den beiden Programmen hin- und herspringen. Jedes Programm ist dabei völlig unabhängig von dem anderen und kann über eine eigene Liste an Banken und Programm-Schirmen verfügen.

Jetzt wissen wir, wie das Ganze im Prinzip funktioniert. Es geht aber nicht nur mit zwei, sondern mit beliebig vielen Programmen. Sie können die Programme einzeln über die Optionen *St. Andere und Ed Andere* im Menü-Fenster aufgerufen werden. Sie erhalten dann einen besonderen Programm-Selektor, der dem AMOS Datei- Selektor beinahe aufs Haar gleicht. Der einzige Unterschied besteht darin, daß Sie unter den Programmen im Speicher und nicht auf der Diskette wählen können. Das Programm wählen Sie mit der Maus und klicken es dann mit der linken Maustaste an.

Versuchen Sie doch, Programm 1 und Programm zwei mit diesem System zu bearbeiten und laufen zu lassen. Sie werden sehen, es dauert nicht lange, bis Sie es im Schlaf können. Weitere Hinweise hierzu finden Sie auch bei den Befehlen *Andere Lade und Andere Neu.*

Zusätze

Damit man die Zusätze (Accessories) leichter von den normalen Basic-Programmen unterscheiden kann, erhalten Sie die Erweiterung ".ACC" anstatt des üblichen ".AMOS". Mit dem Befehl *Andere Lade* können die Zusätze wie jedes andere Programm in den Speicher geladen werden. Sie sehen dann einen normalen Datei- Selektor, über die Sie den Zusatz von der Diskette laden können. Wenn der Zusatz dann im Speicher ist, können Sie direkt in Ihr aktuelles Programm zurückgehen und den Zusatz über die Option St. Andere im Menü-Fenster jederzeit aufrufen. Gehen Sie nur einfach mit dem Mauszeiger auf den gewünschten Zusatz und drücken Sie die linke Taste.

Oder Sie können alle Zusätze auf der Diskette mit der Option *Ac.Neu/Lade* gleichzeitig laden. Diese Option finden Sie im SYSTEM-Menü, das angezeigt wird, wenn Sie die rechte Maustaste drücken. Aber beachten Sie, daß Ac.Neu/Lade alle bisher geladenen Zusätze löscht und durch die Zusätze auf der Diskette, die jetzt im Laufwerk ist, ersetzt.

Überzeugen Sie sich selbst: stecken Sie die AMOS-Programmdiskette ins Laufwerk und wählen Sie *Ac.Neu/Lade* aus dem SYSTEM-Menü. Dann wird schnell der Hilfe-Zusatz geladen. Das Besondere daran ist, daß man den Hilfe-Text direkt durch Drücken der HELP-Taste abrufen kann. Wir haben alles hineingepackt, was Sie über die Zusatzprogramme wissen müssen. Sie brauchen nur noch den auf dem Bildschirm angezeigten Anweisungen zu folgen.

Der Direktmodus

Aus dem Editier-Fenster kann man jederzeit durch Drücken von Escape in den Direktmodus gehen. Als Default wird das Direkt-Fenster in der unteren Hälfte des

Bildschirms gezeigt, mit dem Programm-Schirm im Hintergrund. Wenn Sie jetzt ein Programm ablaufen lassen, das das Bildschirmformat ändert, Fenster zeigt, Sprites animiert usw., dann bleiben all diese Daten unverändert. Sie können also das Direkt-Fenster verschieben oder in den Editor zurückgehen, um das Programm zu verändern, ohne die aktuelle Bildschirmanzeige zu zerstören. Dieses Direkt-Fenster ist völlig eigenständig und wird auf seinem eigenen vorderen Bildschirm-Niveau angezeigt.

Im Direktmodus können Sie jede beliebige AMOS Basic-Zeile eingeben. Die einzigen Befehle, die Sie nicht anwenden können, sind Anweisungen für Schleifen (Loops) oder Zweige (Branch). Sie haben nur Zugang zu den normalen Variablen (im Unterschied zu den lokalen Variablen, die in einer Prozedur definiert sind).

Die Editier-Tasten im Direktmodus

ESCape	Sprung ins Editier-Fenster
ENTER	Ausführen der Befehle in dieser Zeile
DELete	Löscht Zeichen unter dem Cursor
Backspace	(Rücktaste) Löscht Zeichen links vom Cursor
Pfeil links	Cursor nach links
Pfeil rechts	Cursor nach rechts
Shift+links	Ein Wort nach links
Shift+rechts	Ein Wort nach rechts
Shift DELete	Löscht ganze Zeile
Shift BACK	dto.
Help	Zeigt die Definitionen der Funktionstasten an
F1 bis F10:	Diese Tasten speichern die letzten zehn Zeilen, die Sie im Direktmodus eingegeben haben. F1 ist die letzte, F2 die vorletzte usw. Wenn Sie ins Editier-Fenster zurückgehen, oder ein Programm ablaufen lassen, wird dieser Speicher gelöscht.

Das Menü-Fenster

Wir geben Ihnen hier eine detaillierte Erklärung aller Optionen, die Sie vom Menü-Fenster aus wählen können.

Default-Menü

Über dieses Menü erhalten Sie Zugang zu den verschiedenen Befehlen, die Sie im Editor einsetzen, sowie zu den Block- und Such-Menüs.

Start **(F1)**
Das derzeit im Speicher befindliche Programm läuft ab.

Test **(F2)**
Überprüft die Syntax des gesamten Programms und stellt den Cursor auf den ersten Fehler.

Einrücken **(F3)**
Rückt das Listing des aktuellen Programms für Sie ein.

Block-Menü **(Ctrl oder F4)**
Zeigt das Block-Menü im Auswahlfenster an. Diese Optionen können jetzt entweder mit der Maus oder der entsprechende Funktionstaste abgerufen werden. Durch Klicken mit der rechten Maustaste, Entfernen des Maus-Cursors aus dem Bereich der Funktionstasten oder Drücken einer Taste können Sie ins Hauptmenü zurückgehen.

Such-Menü **(Alt oder F5)**
Mit diesem Such-Menü können Sie in Ihrem Programm nach bestimmten Worten suchen und sie auch ersetzen. Von diesem Menü aus können Sie auch den Umfang des Text-Puffers oder den Tabulator ändern.

St.Andere **(F6)**
Ruft ein Programm oder einen Zusatz aus dem Speicher ab.

Ed.Andere **(F7)**
Editiert ein Programm, das über die Befehle *Ac.Neu/Lade oder Andere Lade* im Speicher installiert wurde. Wenn Sie Ihr aktuelles Programm nicht gespeichert haben, werden Sie dazu aufgefordert, dem Werk einen Namen zu geben, denn es wird jetzt in den Speicher geschoben.

Sie können über den Datei-Selektor nun ein anderes Programm zum Bearbeiten wählen. Dazu gehen Sie nur mit der Maus auf den entsprechenden Dateinamen und drücken die linke Taste.

Beachten Sie bitte, daß der Umfang des Editor-Puffers zusammen mit Ihrem Programm gespeichert wird. Wenn Sie Ihr Programm das nächste Mal bearbeiten, nimmt der Puffer wieder seine ursprüngliche Größe an. Wollen Sie ein Programm bearbeiten, das nicht auf diese Art und Weise gespeichert wurde, wie zum Beispiel ein Zusatz, dann wird der Puffer automatisch vergrößert, wenn das erforderlich ist. Ist der Speicherplatz verbraucht, wird die Option abgebrochen, und die Fehlermeldung "Kein Speicher mehr vorhanden" erscheint.

Überschreib **(F8)**
Wechselt zwischen zwei verschiedenen Editier-Modi ab.

Einfügen Fügt in den bestehenden Text jedes Zeichen, das Sie schreiben, ein und schiebt den Text nach dem Einschub unverändert weiter (Default). Sie editieren z.B. folgende Zeile:

```
Rem TESTET DEN INSRT-MODUS
```

Der Cursor steht unter dem R. Sie fügen ein E ein, und die Zeile lautet dann:

Rem TESTET DEN INSERT-MODUS

Überschreiben Ersetzt das Zeichen, auf dem der Cursor steht. Beim obigen Beispiel sieht das dann so aus:

Rem TESTET DEN INSET-MODUS

Sie können Überschreiben nach Belieben ein- und ausschalten, indem Sie die Option nochmals anwählen. Der aktuelle Modus wird jeweils durch das Zeichen ganz links in der Status-Zeile angezeigt (I=Insert; O=Overwrite). Wenn Sie beim Einfügen einen Fehler machen, dann können Sie die Änderungen in der aktuellen Zeile mit Ctrl+U normalerweise rückgängig machen.

Falten (F9)

Mit dieser Option wird eine Prozedur-Definition in Ihrem Programm-Listing verborgen, und nur die erste Zeile der Definition im Listing angezeigt. Dazu geht man mit dem Cursor an eine beliebige Stelle der Definition und wählt die Option Falten. Dann wird die Prozedur auf Syntax-Fehler überprüft und im Programm-Listing versteckt. Dabei wird nichts verändert, sondern nur die Anzeige dieser Zeilen im Listing sieht jetzt anders aus.

Normalerweise macht man dies rückgängig, indem man mit dem Cursor auf eine verborgene Prozedur geht und nochmals **Falten** anklickt (oder F9 drückt). Wenn Sie der Sache nicht so ganz trauen, so können Sie zur zusätzlichen Sicherung den LOCK-Zusatz von der AMOS-Programmdiskette aufrufen.

- Erstellen Sie eine Sicherungskopie Ihres Programms mit **geöffneten** Prozeduren.
- Schließen Sie alle Prozeduren, die Sie ganz bestimmt verschließen möchten.
- Laden Sie das Programm FOLD.AMOS als Zusatz.
- Lassen Sie FOLD.AMOS über die Menüoption St. Andere aus dem Editor ablaufen.
- Nun sind Ihre Prozeduren verschlossen, und Sie können sie nicht wieder öffnen.

Das Schöne an diesem System ist, daß Sie ganze Libraries mit Ihren Routinen auf der Diskette erstellen können. Diese können dann als separates Programm in den Speicher geladen werden (siehe LOAD OTHER). Dann können Sie die Routine, die Sie brauchen, ausschneiden und direkt in Ihr Hauptprogramm kopieren. Und so können Sie eine Routine, die Sie einmal geschrieben haben, einsetzen so oft Sie wollen.

Wenn Sie dieses System anwenden möchten, dann sollten Sie verschiedene Punkte berücksichtigen:

- Wenn Sie eine Prozedur verbergen bzw wieder anzeigen, dann wird automatisch die Syntax des **gesamten** Programms überprüft. Ist ein Fehler vorhanden, so wird der Befehl nicht ausgeführt. Deshalb sollten Sie immer eine Sicherungskopie Ihrer Prozeduren in voller Länge, also im "Aufgefalteten"-Format haben.

- Versuchen Sie nicht, eine verborgene Prozedur mit den normalen Cursor-Tasten zu löschen, denn das hat keine nachhaltige Wirkung. Stattdessen sollten Sie die Prozedur als Block markieren und den Bl. Schneide-Befehl verwenden.

- Die Befehlee *Bl. Schneide / Bl. Einfügen* können problemlos auch bei verborgenen Prozeduren angewendet werden. Die gesamte Prozedur-Definition wird gespeichert, wenn Sie die Prozedur-Anweisung mit *Bl. Schneide* bearbeiten. Aber Sie sollten versuchen, die folgenden Fehler zu vermeiden:

"Dieses Feld wurde im Hauptprogramm nicht definiert"

Wenn man eine Routine von einem Programm in ein anderes kopiert, vergißt man leicht die im Hauptprogramm definierten SHARED oder GLOBAL Variablen und Parameter. Wenn eine Prozedur auf externe Variablen zugreift, die nicht definiert sind, erhalten Sie die oben genannte Fehlermeldung. Also überprüfen Sie am besten Ihr ursprüngliches Listing auf SHARED- oder GLOBAL-Anweisungen.

"Sprungmarke doppelt definiert"

Sie haben versucht, einen Ablauf ZWEIMAL im selben Programm zu kopieren. Sie haben wahrscheinlich eine zusätzlichen Prozedur aus Versehen miterwischt.

"Procedure nicht definiert"

In AMOS-Basic ist es völlig in Ordnung, Prozeduren ineinander zu schachteln. Es kann nur manchmal zu Problemen kommen, wenn Sie versuchen, diese Prozeduren wieder auseinander zu sortieren. Das kann nämlich nur gelingen, wenn auch sämtliche verwendeten Prozeduren in das Programm kopiert wurden. Daher ist es ratsam, alle diese Routinen am Anfang einer Prozedur aufzuführen, denn das kann eine Menge Verwirrung sparen!

Zeile einf. (Ctrl+I oder F10)
Fügt an der aktuellen Cursor-Position eine Zeile ein.

Das System-Menü

Das SYSTEM-Menü enthält eine Reihe von Befehlen, die es Ihnen ermöglichen, Ihre Programme von der Diskette zu laden und dort auch wieder zu speichern. Man ruft dieses Menü mit der Shift- (Umschalt-)Taste oder durch Drücken der rechten Maustaste auf. Und hier ist eine vollständige Liste der verfügbaren Optionen.

Laden (Shift+F1 oder Amiga+L)
Lädt eine AMOS Basic-Datei von Diskette. Die Datei wird über den AMOS Datei-Selektor ausgesucht.

Sichern (Shift+F2 oder Amiga+S)

Speichert das aktuelle AMOS-Programm. Wenn Sie eine Datei zum ersten Mal speichern, dann müssen Sie über den Datei-Selektor den Namen eingeben. Ist auf der Diskette bereits ein anderes Programm mit dem gleichen Namen, so wird dieses automatisch durch Hinzufügen der Erweiterung ".BAK" umbenannt. Das ist ein guter Schutz vor Fehlern, denn Sie können so meist die vorherige Version Ihres Programms zurückholen, falls Sie tatsächlich Quatsch gemacht haben.

Sichern als (Shift+F3 oder Shift+Amiga+S)

Speichert das laufende Programm unter einem anderen Namen. Der neue Name wird auch mit Hilfe des Datei-Selektors ausgesucht. Achtung: wenn Sie ein Programm "AUTOEXEC.AMOS" nennen, dann wird es beim Einschalten automatisch geladen und läuft dann ab. Wenn Sie in AMOS-Basic gehen, dann erscheint sofort der Schirm des Hauptprogramms.

Kombinieren (Shift+F4)

Fügt das ausgewählte Programm bei der aktuellen Cursor-Position ein, ohne Ihr ursprüngliches Programm vorher zu löschen. Dadurch können Sie Routinen aus einem anderen Programm, wie zum Beispiel dem AMOS Map Definer, einbauen.

Komb.ASCII (Shift+F5)

Verbindet die Ascii-Version eines AMOS Basic-Programms mit dem aktuellen Programm im Speicher.

Ac.Neu/Lade (Shift+F6)

Löscht alle aktuellen Zusatz-Routinen aus dem Speicher und ersetzt sie durch die Routinen auf der Diskette im Laufwerk. So werden alle Dateien mit der Erweiterung ".ACC" automatisch geladen. Wenn Sie auf einem A500 ohne Speichererweiterung arbeiten, sollten Sie diesen Befehl mit etwas Vorsicht genießen, denn sonst kann der Speicher knapp werden.

Andere Lade (Shift+F7)

Lädt ein einziges Zusatz-Programm von der Diskette im Laufwerk. Sie erhalten Zugang über den Befehl "Andere Lade" im Hauptmenü.

Andere Neu (Shift+F8)

Löscht ein oder mehrere Zusatz-Programm(e) aus dem Amiga-Speicher. Wenn Sie diesen Befehl aufrufen, dann erscheint der AMOS Datei-Selektor, und Sie zeigen durch Anklicken an, welche(s) Programm(e) Sie löschen möchten, oder wählen die **Alle-**Taste, und löschen alle Zusatz-Programme.

Neu (Shift+F9)

Löscht das Programm, das Sie gerade bearbeiten. Wenn Sie das Programm noch nicht gespeichert haben, können Sie es noch auf Diskette speichern, bevor es gelöscht wird.

Ende (Shift+F10)
Steigt aus AMOS aus und geht zurück zu CLI. Wie bei NEU haben Sie die Möglichkeit, Ihr Programm vorher noch zu speichern.

Das Block-Menü

Das Block-Menü ermöglicht Ihnen das Verschieben von ganzen Abschnitten Ihres Programms. Sie können die Optionen auch direkt mit der Maus aus dem Hauptmenü aufrufen. Aber Sie werden wahrscheinlich mit dem Abrufen über die Tastatur schneller sein. Hier sind alle Optionen:

Bl.Anfang (Ctrl+B oder Ctrl+F1)
Anfangspunkt des aktuellen Blocks.

Block Ende (Ctrl+E oder Ctrl+F6)
Definiert das Ende eines Blocks. Normalerweise folgt dieser Befehl direkt auf Ctrl+B bzw. Ctrl+F1. Jetzt wird der Bereich des Blocks hervorgehoben.

Bl. Schneide (Ctrl+C oder Ctrl+F2)
Nimmt den ausgewählten Block von der aktuellen Position und lädt ihn in den Speicher. Jetzt können Sie den Block überall in Ihrem Programm mit dem *Bl.Einfügen*- Befehl wieder einsetzen.

Bl. Einfügen (Ctrl+P oder Ctrl+F7)
Fügt den gesamten Inhalt des Blocks an der aktuellen Cursor-Position wieder ein. Der Block muß allerdings zuvor mit dem Bl.Schneide- oder Bl.Merken-Befehl gespeichert worden sein.

Bl. Schiebe (Ctrl+M oder Ctrl+F3)
Bewegt den markierten Block zur aktuellen Cursor-Position, wobei die ursprüngliche Version gelöscht wird.

Bl. Merken (Ctrl+S oder Ctrl+F8)
Kopiert den Inhalt eines Blocks in den Speicher ohne Auswirkung auf das Programm. Das ist eine einfache Art, um Information zwischen den Programmen im Speicher zu übertragen.

Bl. Versteck (Ctrl+H oder Ctrl+F4)
Hebt die mit Bl.Anfang und Block Ende gesetzten Markierungen wieder auf.

Bl. Sichern (Ctrl+F9)
Speichert den markierten Block als AMOS-Programm auf der Diskette. Sie können ihn jetzt mit den KOMBINIERE- oder LADE- Befehlen aus dem SYSTEM-Menü wieder laden. Achtung: die Speicherbanken werden nicht zusammen mit Ihrem Programm-Listing gespeichert.

Sichern ASCII (Ctrl+F5)
Speichert den gewählten Block als normale Text-Datei auf der Diskette. Diese Datei kann dann direkt in jedes Standard-Textverarbeitungsprogramm geladen werden. Wenn Sie Zugang zu einem Modem haben, können Sie diese Dateien auch in Bulletin-Boards und Kommunikationsnetze wie MicroLink und Prestel stellen.

Bl. Drucken (Ctrl+F10)
Der Block wird sofort über den - angeschlossenen! - Drucker ausgedruckt.

Außerdem kann man über die Tastatur mit **Ctrl+A** einen besonderen *Wähle Alles* Befehl aufrufen, damit wird das ganze Programm als Block erfaßt. Das ist zum Beispiel sehr nützlich, wenn Sie das gesamte Programm als Ascii-Datei speichern wollen.

Das Such-Menü

Sie können sich wohl mit am besten mit dem AMOS-System vertraut machen, indem Sie die Beispiel-Programme auf der AMOS Daten-Diskette unter die Lupe nehmen. Mit Hilfe das Such-Menüs können Sie zum Beispiel die Listings nach jeder beliebigen AMOS-Anweisung absuchen. So erhalten Sie wertvolle Tips, wie Sie sie im Zusammenhang mit einem echten Programm einsetzen können.

Sie können auch mit dem *Ersetzen*-Befehl die Namen der Variablen in einem Ihrer Basic Programme ändern. So können Sie einfache, kurze Namen verwenden, wenn Sie Ihre Programme eingeben, und sie dann am Ende gegen besser verständliche austauschen.

Das SUCH-Menü kann direkt aus dem Menü-Fenster mit der Maus oder über die entsprechenden Tastenkombinationen aufgerufen werden.

Suchen (Ctrl+F oder Alt+F1)
Gibt eine Folge von bis zu 32 Zeichen über die Tastatur ein und sucht den Text ab, bis die exakte Entsprechung gefunden ist. Die Suche erfolgt von der aktuellen Cursor-Position nach unten.

Weiter S. (Ctrl+N oder Alt+F2)
Sucht nach dem nächsten Vorkommen der durch *Suchen* eingegebenen Zeichenfolge.

Vorne S (Alt+F3)
Identisch zum *Suchen-Befehl;* die Suche beginnt jedoch nicht bei der aktuellen Cursor-Position, sondern am Anfang des Programms.

Ersetze (Alt+F4 oder Ctrl+R)
Die Wirkung dieses Befehls hängt davon ab, wann er benutzt wird. Es gibt zwei Möglichkeiten:

- **Vor** einem Suchen-Befehl: dann müssen Sie jetzt eine Zeichenfolge eingeben
- **Nach** einem Suchen-Befehl: ist die Suche erfolgreich gewesen, wird der Text durch die

eingegebene Zeichenfolge ersetzt. *Ersetze* bringt Sie nun zum nächsten Vorkommen der Suchfolge in Ihrem Programm. Wenn Sie sie hier nicht ersetzen wollen, dann können Sie sie mit *Weiter S.* überspringen.

Ersetze alle (Alt+F5)
Ersetzt jedes Vorkommen der Zeichenfolge in Ihrem Programm. Sie gehen folgendermaßen vor:

- Bestätigen Sie den Befehl durch Eingeben von "J" über die Tastatur oder Anklicken des "JA" Feldes in der Status-Zeile.
- Geben Sie die Zeichenfolge ein, die Sie verändern möchten.
- Geben Sie die Zeichenfolge ein, die sie ersetzen soll. Jetzt beginnt das Suchen und Ersetzen dieser Zeichenfolge vom Anfang Ihres Programms an.

Klein<>Groß (Alt+F6)
Der Default ist so eingestellt, daß alle Kleinbuchstaben von den entsprechenden Großbuchstaben beim Suchen und Ersetzen unterschieden werden. So werden also die Buchstaben "g" und "G" bei der Suche wie unterschiedliche Buchstaben behandelt. Diese Option setzt Klein- und Großbuchstaben gleich, und die neue Anzeige lautet Klein=Groß.

Alle öffnen (Alt+F7)
Öffnet alle geschlossenen Prozeduren in Ihrem Programm. Die Syntax des gesamten Programms wieder überprüft, bevor die Prozeduren wieder angezeigt (aufgefaltet) werden. Wird dabei ein Fehler festgestellt, so führt das zum Abbruch.

Sollten Sie mit diesem Befehl Schwierigkeiten haben, schlagen Sie bitte unter dem *Falten*- Befehl nach, dort finden Sie Einzelheiten über mögliche Probleme und ihre Lösung.

Alle zu (Alt+F8)
Schließt alle Prozedur-Definitionen in Ihrem aktuellen Programm. Nur die erste Zeile Ihrer Prozedur-Definition wird im Listing auftauchen. Dadurch werden die Listings viel kürzer und übersichtlicher. Wie der oben aufgeführte *Öffnen*-Befehl oben, kann auch dieser Befehl nur ausgeführt werden, wenn die Syntax des Programms keine Fehler aufweist. Wenn Sie die Prozeduren geschlossen haben, können Sie sie über den *Falten*-Befehl einzeln überarbeiten.

Textpuffer (Alt+F9)
Einstellen des Text-Puffers. Verändert die Anzahl der verfügbaren Zeichen für Ihre Listings. Sie können so die Speicherkapazität des Editors erweitern, wenn Sie besonders große Programme in Ihren Amiga laden wollen. Dieser Befehl ist ganz besonders nützlich, wenn Sie eine Speichererweiterung installiert haben. Siehe auch CLOSE EDITOR.

Tabulator (Alt+F10)
Legt die Anzahl der Zeichen zwischen den Tabulator-Stops fest.

Tastatur-Makros

Mit AMOS-Basic können Sie bis zu 20 Tastatur-Makros gleichzeitig erstellen. Das Abrufen erfolgt über eine Tastenkombinination der linken oder rechten Amiga- und einer Funktionstaste. Wenn Sie einmal ein Makro definiert haben, können Sie es überall in AMOS-Basic einsetzen, so als würden Sie die Befehle direkt über die Tastatur eingeben. Ein und dasselbe Makro kann im Editier-Fenster, im Direktmodus, und sogar aus einem Ihrer Basic-Programme heraus aufgerufen werden.

Die aktuelle Belegung der Tasten wird in einem speziellen Bereich über dem Direkt-Fenster angezeigt. Wenn Sie im Direktmodus sind, können Sie diese Liste mit der HILFE-Taste auf den Bildschirm rufen. Wenn Sie im Editor sind, rufen Sie die Tastenbelegung mit der rechten oder linken Amiga-Taste auf, und sie wird im Menü-Fenster dargestellt.

Als Default wird die Makro-Tastenbelegung als ein Satz von allgemeinen Basic-Keywords geladen. Sie können mit einer Option aus dem Konfigurations-Zusatz (CONFIG1.3.AMOS) geändert werden. Sie können die Belegung dieser Tasten auch direkt in einem Ihrer Programme mit der starken KEY$-Funktion zuweisen.

=KEY$= *(Definiert ein Tastatur-Makro)*

KEY$(n)=command$
command$=KEY$(n)

KEY$ weist den Inhalt der Zeichenkette *command$* der Funktionstaste *n* zu. *n* ist die Kennzahl der Funktionstasten 1 bis 20. Die Tasten von eins bis zehn werden durch Drücken der linken Amiga-Taste zusammen mit der Funktionstaste aufgerufen, und die Tasten ab elf durch Kombination von rechter Amiga-Taste und den Funktionstasten.
Dabei müssen Sie unbedingt beide Tasten gleichzeitig drücken, sonst wird Ihr Makro als das Drücken zweier getrennter Tasten fehlinterpretiert. *command$* kann jede beliebige Kette von bis zu 20 Zeichen sein.

Es gibt zwei Spezialzeichen, die von dieser Funktion direkt interpretiert werden:

'(Alt+Apostroph) Generiert einen Enter-Code

'(Apostroph) Umschließt einen Kommentar (comment). Das wird nur in Ihrer Tastaturliste angezeigt und von der Makro- Routine völlig ignoriert. Beispiele:

```
Print Key$(1)
Key$(2)="Default'"
Alt+F2
```

L. Amiga + F2

In der Praxis erweist sich das Makro-System meist als ungemein nützlich. Sie können damit nicht nur das Eingeben Ihres Basic-Programms erheblich beschleunigen, sondern auch eine Liste von Standard-Inputs definieren. Sie können anhand des Programms BEISPIEL 3.2 im MANUAL Unterverzeichnis sehen, wie extrem wirksam sie in einem Adventure sind.

Wenn Sie eine Tastenbelegung generieren möchten, die keine Ascii-Entsprechung hat, wie z.B. eine Pfeiltaste, können Sie wahlweise mit der SCAN$-Funktion einen Scan-Code in diesen Makros integrieren.

=SCAN$ *(Erstellt einen Scan-Code für den Einsatz mit KEY$)*

x$=Scan$(n[,m])

n ist der Scan-Code einer Taste, die in einer Ihrer Makro-Definitionen verwendet wird, *m* ist ein wählbare Maske, die die Spezialtasten wie Ctrl oder Alt in das folgende Format bringt:

Bit	Getestete Taste	Bemerkungen
0	Linke Shift-Taste	
1	Rechte Shift-Taste	
2	Feststelltaste (Caps Lock)	Entweder EIN oder AUS
3	Control (Ctrl)	
4	Linke Alt-Taste	
5	Rechte Alt-Taste	
6	Linke Amiga-Taste	Auf manchen Tastaturenist das die Commodore- Taste
7	Rechte Amiga-Taste	

Wenn ein Bit auf eins gesetzt ist, dann wird die entsprechende Taste in Ihrem Makro "gedrückt". Zum Beispiel:

```
KEY$(4)="Uii!"+Scan$($4C)
KEY$(5)="Seite rauf!"+scan$($4c,%00010000)
```

Speicherverwaltung

Wenn Sie auf einem A500 ohne Speichererweiterung arbeiten, dann kann es manchmal etwas kneifen. Deshalb bieten wir Ihnen zwei ganz starke Anweisungen, mit denen Sie den verfügbaren Speicherplatz maximal nutzen können.

Bevor wir diese Funktionen besprechen, wollen wir Sie darauf aufmerksam machen, daß Sie zum Beispiel schon einmal ungefähr 30KB sparen können, wenn Sie Ihr externes 3.5 Zoll-Laufwerk außer Betrieb setzen, bevor Sie AMOS-Basic laden.

Da AMOS auf die Diskette nur für kleine Library-Dateien zugreift, werden Sie feststellen, daß Sie das zweite Laufwerk so gut wie nie brauchen. Also schnappen Sie

sich den Speicher!

ACHTUNG! Niemals das Laufwerk ausschalten, wenn der Amiga eingeschaltet ist, denn das bringt überhaupt nichts. Der Speicher wird dem Laufwerk nämlich bei der Inbetriebnahme zugeteilt.

CLOSE WORKBENCH *(Schließt Workbench)*

CLOSE WORKBENCH

Dieser Befehl schließt den Workbench-Schirm und spart so ungefähr 40KB Speicherkapazität. Zum Beispiel:

```
Print Chip Free,Fast Free
Close Workbench
Print Chip Free,Fast Free
```

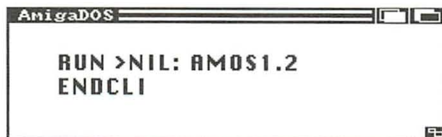
Sie können CLOSE WORKBENCH entweder aus dem Direktmodus oder in einem Ihrer Basic Programme aufrufen. Ein typische Programmzeile sieht dabei etwa so aus:

```
If Fast Free=0 Then Close Workbench
```

Daraufhin wird der Speicher überprüft, und wenn kein zusätzlicher Speicherplatz verfügbar ist, wird Workbench geschlossen.

Schließen der Workbench

Beachten Sie bitte, daß der Befehl CLOSE WORKBENCH keine Wirkung zeigt, wenn Sie ein CLI-Fenster geöffnet haben. Wollen Sie die Workbench tatsächlich schließen und anfangen, mit AMOS zu arbeiten, so müssen Sie die folgenden CLI- Anweisungen verwenden:



CLOSE EDITOR *(Schließen des Editier-Fensters)*

CLOSE EDITOR

Schließt das Editier-Fenster während Ihr Programm abläuft. Damit sparen Sie mehr als 28KB göttlichen Speicher. Und darüber hinaus hat es überhaupt keine Nebenwirkungen auf Ihr Programm-Listing!

Wenn Ihr Programm dann fertigabgelaufen, aber nicht genug Speicher vorhanden ist, um das Fenster wieder zu öffnen, löscht AMOS einfach Ihren aktuellen Schirm und kehrt zum Standard-Defaultschirm zurück. Mit der Escape-Taste springen Sie jetzt einfach wie gehabt in den Editor zurück. So, jetzt sind Sie aber platt!

Integrierte Zusätze

Jetzt wollen wir einmal die Techniken unter die Lupe nehmen, die Sie zum Schreiben Ihrer eigenen Zusatzprogramme benötigen. Eigentlich handelt es sich dabei nur um eine besondere Art der verschiedenen Programme, die wir schon etwas früher angesprochen haben. Wie erwartet, können sie alle üblichen Basic- Anweisungen beinhalten.

Zusätze (Accessories) werden direkt über Ihrem aktuellen Programm-Schirm angezeigt, und Musik, Sprite oder Bob-Animation werden automatisch vom Bildschirm entfernt.

Deshalb sollte Ihr Zusatz-Programm die Art und Maße des Bildschirms in der Installationsphase mittels der Befehle SCREEN HEIGHT, SCREEN WIDTH und SCREEN COLOUR überprüfen. Wenn Ihr aktueller Schirm nicht paßt, kann es sein, daß Sie für das Zusatz-Fenster einen neuen Schirm aufrufen, oder alle bestehenden Schirme mit einer DEFAULT-Anweisung löschen müssen.

Alle von Ihrem Zusatz-Programm benutzten Speicherbanken sind vom Hauptprogramm völlig unabhängig. Sie können die Banken mit einem besonderen BGRAB-Befehl ändern.

BGRAB *(Erfaßt die vom aktuellen Programm benutzten Banken)*

BGRAB b

BGRAB "borgt sich" eine Bank von dem aktuellen Programm und kopiert sie in die gleiche Bank in Ihrem Zusatz-Programm. Wenn die Bank für den Zusatz bereits existiert, dann wird sie total gelöscht. Wenn der Zusatz zum Editor zurückkehrt, dann wird die Bank, die Sie gegriffen haben, automatisch zusammen mit allen vorgenommenen Veränderungen in das Hauptprogramm zurückgebracht. b ist hierbei die Nummer einer Bank von 1 bis 16.

Beachten Sie bitte, daß dieser Befehl nur innerhalb eines Zusatz-Programms angewendet werden kann. Wenn Sie versuchen, ihn in einem normalen Programm einzusetzen, werden Sie die entsprechende Fehlermeldung erhalten.

PRUN *(Ablaufen eines Programms aus dem Speicher)*

PRUN "Name"

Läßt ein Basic-Programm ablaufen, das zuvor im Speicher des Amiga installiert wurde. Dieser Befehl kann entweder im Direktmodus oder in einen Programm eingesetzt werden. Eigentlich ist PRUN einem normalen Ablauf-Befehl sehr ähnlich, abgesehen davon, daß alle BOB-, Sprite- und Musikfunktionen völlig unterbunden sind.

Beachten Sie hier, daß es nicht möglich ist, dasselbe Programm zweimal während einer Arbeitssitzung ablaufen zu lassen. Wenn Sie es einmal aufgerufen haben, werden alle weiteren Versuche völlig ignoriert.

Wenn das Programm in Ihr Zusatz-Programm zurückkehrt, werden Sie Ihren Schirm wieder in den ursprünglichen Zustand versetzen müssen. So wird die Gefahr vermieden, daß die Schirme Ihres Zusatz-Programms durch die neue Routine zerstört werden. Siehe dazu auch BEISPIEL 3.3 im MANUAL- Unterverzeichnis.

=PRG FIRST$ *(Liest das erste Programm, das in den Speicher geladen wurde)*

p$=PRG FIRST$

So erhält man die Bezeichnung des ersten Basic-Programms, das im Amiga- Speicher installiert wurde. Dieser Befehl wird in Verbindung mit dem folgenden PRG NEXT$ Befehl eingesetzt, um eine vollständige Liste aller gegenwärtig verfügbaren Programme zu erstellen.

=PRG NEXT$ *(Nennt das nächste im Speicher installierte Programm)*

p$=PRG NEXT$

PRG NEXT$ wird nach dem PRG FIRST$ Befehl eingesetzt, damit Sie durch alle im Amiga-Speicher installierten Programme blättern können. Wenn Sie das Ende der Liste erreichen, wird ein Wert von "" von dieser Funktion angegeben. Hier ist ein Beispiel:

```
N$=Prg First$
While N$<>""
  Print "Programm";N$
  N$=Prg Next$
Wend
```

=PSEL$ *(Ruft Programm-Selektor auf)*

N$=PSEL$("filter"[default$,title1$,title2$])

PSEL$ ruft einen Programm-Selektor auf, der zu den von St. Andere, Ed.Andere, Lade Andere und Neu Andere aufgerufenen Verzeichnissen identisch ist. Dann kann wie gewöhnlich ein Programm ausgesucht werden, dessen Name in n$ eingegeben wird. Wenn Sie aus dem Programm-Selektor aussteigen, wird anstatt n$ nur ein leeres "" angezeigt.

"filter" bestimmt die Art der Programme, die zur Auswahl angezeigt werden. Dafür kann z.B. eingesetzt werden:

"*.ACC"	Liste aller Zusatz-Programme im Speicher
"*.AMOS"	Zeigt nur die installierten AMOS-Programme an
"*.*"	Zeigt alle zur Zeit im Speicher befindlichen Programme an

Weitere Einzelheiten dazu finden Sie beim DIR-Befehl.

default$	enthält den Namen des Programms, das als Default eingesetzt wird
title1$,title$	enthält bis zu zwei Zeilen Text, die oben im Selektor angezeigt werden

Eine Demonstration dieser Anweisung erhalten Sie in **BEISPIEL 3.4** im MANUAL-Unterverzeichnis.

Das HELP-Zusatz-Programm

Wenn in Editier-Fenster die HILFE-Taste gedrückt wird, greift AMOS automatisch auf ein Hilfe-Programm namens HELP.ACC zu, sofern es im Speicher ist. Im Gegensatz zu den anderen Zusätzen wird dieses Programm ~~wird~~ direkt über dem Editier-Fenster angezeigt. Das Wort, das Sie gerade bearbeiten, ist zugänglich, und die Zeilen-Adresse dieses Wortes wird in einem Adressen-Register gespeichert, und kann über die AREG-Funktion dann gelesen werden.

Die Steuerungstasten im Editor

Zum Abschluß dieses Themas hier nun eine vollständige Liste der verschiedenen Steuerungstasten und ihrer Wirkung.

Spezialtasten

ESCape	Bringt Sie in den Direktmodus

Editier-Tasten

Backspace (Rücktaste)	Löscht das Zeichen links vom Cursor.
DELete (Löschtaste)	Löscht das Zeichen unter dem Cursor.
ENTER	Eingabe der aktuellen Zeile. Wenn Sie in eine Zeile gehen und ENTER drücken, wird die Zeile geteilt (aber nur, wenn Sie keine Änderungen gemacht haben.
Shift+Back oder Ctrl+Y	Löscht aktuelle Zeile und zieht dann den Rest des Textes nach oben.
Ctrl+U	Zurück. Ruft beim Überschreiben die letzte Zeile zurück.
Ctrl+Q	Löscht alle restlichen Zeichen in dieser Zeile von der Cursor-Position an.

Ctrl+I	Fügt bei der Cursor-Position eine Zeile ein.

Die Pfeiltasten

Links	Ein Zeichen nach links.
Rechts	Ein Zeichen nach rechts.
Nach oben	Eine Zeile nach oben (nicht, wenn Sie am Anfang Ihres Programmes stehen).
Nach unten	Eine Zeile nach unten.
Shift+links	Cursor auf vorheriges Wort.
Shift+rechts	Cursor auf nächstes Wort.
Shift+oben	Cursor springt in die erste Zeile dieser Seite.
Shift+unten	Cursor springt in die letzte Zeile dieser Seite.
Ctrl+oben	Vorhergehende Seite.
Ctrl+unten	Nächste Seite.
Shift+Ctrl+oben	Textanfang .
Shift+Ctrl+unten	Textende.
Amiga+oben	Scrollt den Text nach oben ohne den Cursor zu bewegen.
Amiga+unten	Scrollt den Text nach unten ohne den Cursor zu bewegen.
Amiga+links	Scrollt das Programm nach links. Der Cursor bleibt unverändert stehen.
Amiga+rechts	Bewegt den Text nach rechts.

Programm-Steuerung

Amiga+S	Speichert Ihr Programm unter einem neuen Namen.
Amiga+Shift+S	Speichert Ihr Programm unter demselben Namen.
Amiga+L	Lädt ein Programm.
Amiga+P	Schiebt das aktuelle Programm in den Speicher und schafft ein neues Programm.
Amiga+F	Schaltet zwischen zwei Programmen im Speicher hin und her.
Amiga+T	Zeigt das nächste Programm im Speicher an. Wenn Sie diese Tastenkombination wiederholen, können Sie alle derzeit im Speicher befindlichen Programme sehen.

Ausschneiden und Ersetzen

Ctrl+B	Markiert Anfang des Blocks.
Ctrl+E	Markiert Endes des Blocks.
Ctrl+C	Block ausschneiden. Speichert den Block und löscht ihn gleichzeitig von seiner derzeitigen Position. Diese Tasten- kombination funktioniert auch in Direktmodus).
Ctrl+M	Bewegt den Block.
Ctrl+S	Speichert den Block, aber löscht ihn nicht gleichzeitig.

Ctrl+P	Setzt den Block an der aktuellen Cursor-Position wieder ein.
Ctrl+H	Versteckt den Block. Die Markierung desgewählten Blocks wird wieder aufgehoben.

Markierungen

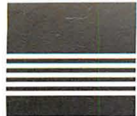
Ctrl+Shift+n	Setzt eine Markierung auf die aktuelle Cursor-Position. *n* muß dabei eine Zahl von 0 bis 9 aus der Zahlentastatur sein.
Ctrl+n	Springt auf eine zuvor gesetzte Markierung.

Suchen/Ersetzen

Alt+oben	Sucht Ihr Programm rückwärts nach der nächsten Zeile mit der definierten Zeichenfolge ab. Wenn AMOS am Ende Ihrer Prozeduren angekommen ist, bleibt der Cursor auf der gegenwärtigen Position stehen.
Alt+unten	Sucht im Programm nach unten nach der definierten Zeichenfolge.
Ctrl+F	Befehl Suchen. Hier müssen Sie die Zeichenfolge eingeben, nach der gesucht werden soll. Dann springt der Cursor von der aktuellen Position aus nach unten zum ersten Vorkommen dieser Zeichenfolge.
Ctrl+N	Findet das nächste Vorkommen der Zeichenfolge.
Ctrl+R	Befehl Ersetzen. Wenn Sie diesen Befehl vor Suchen geben, dann müssen Sie eine Zeichenfolge für das Ersetzen eingeben. Wenn Sie die Zeichenfolge aber schon eingegeben und mit *Suchen* gesucht haben, ersetzt Ctrl+R sie durch den neuen Text, und der Cursor springt an die nächste Stelle mit der Zeichenfolge.

Tabulator

Tab	Bewegt die ganze Zeile an der aktuellen Cursor-Position zum nächsten Tabulator- Stop.
Shift+Tab	Bewegt die Zeile zum vorhergehendenTabulator-Stop.
Ctrl+Tab	Stellt den Tabulator-Abstand ein.



4: Grundregeln

In diesem Kapitel wollen wir Sie mit den Grundregeln für das Erstellen von Programmen in AMOS-Basic bekannt machen und Ihnen zeigen, wie Sie Ihren Programmierstil mit Hilfe der AMOS Basic-Prozeduren (Procedures) verbessern können.

Die Variablen

Die Variablen sind die Namen der Speicherbereiche im Computer. In diesen Speicherbereichen werden zum Beispiel die Ergebnisse der im einem Ihrer Programme durchgeführten Berechnungen verwahrt. Wie Sie diese Variablen nennen steht Ihnen völlig frei, und Sie können jede beliebige Folge von Zeichen oder Zahlen verwenden. Es gibt jedoch ein paar Einschränkungen: alle Variablen müssen mit einem Buchstaben anfangen und dürfen nicht mit einer bereits bestehenden AMOS Basic-Anweisung beginnen. Sie können diese Schlüsselwörter aber innerhalb eines Namens problemlos einsetzen. Variablen wie VPRINT oder SCORE sind also ganz in Ordnung. Die Namen dürfen keinen Leerschritt aufweisen. Sollten Sie dennoch einen Leerschritt wünschen oder benötigen, so müssen Sie "_" stattdessen eingeben.

Hier sind ein paar Beispiele:

AWHILE$, HIGH_SCORE, TEST_FLAG, HEIGHT#

Die Namen können bis zu 255 Zeichen lang sein:

A_VERY_LONG_NAME=10:Rem Das ist ok

Wir haben in den folgenden Beispielen die Fehler für Sie unterstrichen:

WHILE$, 5C, MODERN#, TOAD

Arten von Variablen

AMOS-Basic läßt drei Arten von Variablen in Ihren Programmen zu.

Integervariablen

Anders als die meisten anderen Basic-Sprachen geht AMOS davon aus, daß alle Variablen ganzzahlige Werte sind. Integer sind ganze Zahlen wie zum Beispiel 1,3 oder 8. Sie sind ideal zum Speichern der in Ihren Spielen eingesetzten Werte.

Da die Ganzzahlrechnung viel schneller als die normale Fließkommarechnung geht, kann die Verwendung von ganzen Zahlen in Ihren Programmen zu einer drastischen Beschleunigung führen. Jede ganze Zahl wird in vier Bytes gespeichert und kann von -147.483.648 bis zu +147.483.648 reichen.

Hier nun einige Beispiel für Integer-Variablen:

A, NUMBER, SCORE, LIVES

Reelle Zahlen
Auf diese Variablen folgt in AMOS-Basic immer ein #-Zeichen. Reelle Zahlen können Bruchwerte wie 3,1 oder 1,5 enthalten. Sie entsprechen den Standardvariablen in den meisten anderen Basic-Versionen genau. Jede reelle Variable wird in vier Bytes gespeichert und kann zwischen 1E-14 und 1E-15 liegen. Alle Werte sind bis auf sieben Dezimalstellen genau. Beispiele:

P#, NUMBER#, TEST#

String-Variablen
Diese Variablen bestehen auch Zeichenketten (Strings) und enthalten Text statt Zahlen. Sie unterscheiden sich durch das $-Zeichen am Ende von den normalen Variablen. Die Länge Ihres Textes kann 0 bis 65.000 Zeichen betragen. Hier sind einige Beispiele:

NAMENS$, PATH$, ALIEN$

Zuweisung eines Wertes
Es ist ganz einfach, einer Variablen einen Wert zuzuweisen. Sie wählen nur einen passenden Namen und weisen ihn mit dem "=" Zeichen einem Wert zu.

```
VAR=10
```

Damit wird der Variablen VAR der Wert 10 zugewiesen. Je nach Art Ihrer Variable kann sie nun eine Zahl oder eine Reihe von Zeichen umfassen. Wenn Sie Ihrer Variablen eine Zeichenkette zuweisen wollen, dann setzen Sie sie folgendermaßen in Anführungszeichen:

```
A$="Hallo"
```

Beachten Sie das $-Zeichen hinter dem Namen. Es zeigt AMOS, daß die Variable Buchstaben anstatt Zahlen enthält.

Arrays
Jede Liste von Variablen kann mit Hilfe der DIM-Anweisung zu einem Array zusammengefaßt werden.

DIM *(Dimensioniert ein Array)*

DIM var(x,y,z,...)

DIM definiert in Ihrem AMOS Basic-Programm eine Tabelle von Variablen. Diese Tabellen können beliebig viele Dimensionen enthalten, aber jede Dimension ist dabei auf höchstens 65.000 Elemente beschränkt. Das wird anhand eines Beispiels klarer:

```
Dim A$(10),B(10,10),C#(10,10,10)
```

Ein Element des Arrays wird zugänglich, wenn Sie einfach den Namen des Arrays eingeben, gefolgt von den Index-Zahlen. Diese Zahlen werden durch Kommas getrennt und in runde Klammern () gestellt. Beachten Sie, daß die Element-Zahlen in diesem Array bei Null beginnen. Ein Beispiel:

```
Dim ARRAY(10)
ARRAY(0)=10:ARRAY(1)=15
Print ARRAY(1);ARRAY(0)
15 10
```

```
Dim A(10,10)
A(1,10)=1 : A(10,10)=2
Print A(1,10); A(10,10)
⇓
1 2
```

Die Konstanten

Die Konstanten sind einfach die Zahlen oder Zeichenketten, die einer Variablen zugewiesen oder in einer Ihrer Berechnungen eingesetzt werden. Sie heißen Konstanten, weil sie sich in Laufe Ihres Programms nicht verändern. Alle folgende Werte sind Konstanten:

```
1, 42, 3.141, "Hallo"
```

Als Default werden alle numerischen Konstanten wie ganze Zahlen behandelt. Wird einer Integer-Variablen eine Fließkommazahl zugewiesen, so erfolgt vor dem Einsatz die Umrechnung in einen ganzzahligen Wert. Hier sind einige Beispiele:

```
A=3.141:Print A
3
Print 19/2
9
```

Konstanten können auch in binärer oder hexadezimaler Schreibweise eingegeben werden. Binärzahlen werden durch ein vorausgehendes %-Zeichen gekennzeichnet und Hexadezimal-Zahlen durch ein $-Zeichen. Im folgenden Beispiel zeigen wir Ihnen die verschiedenen Arten, auf die die Zahl 255 dargestellt werden könnte:

Dezimal:	**255**
Hexadezimal:	**$FF**
Binär:	**%11111111**

Beachten Sie bitte, daß alle Zahlen, die Sie in Amos Basic eingeben, automatisch in ein spezielles, internes Format umgewandelt werden. Wenn Sie Ihr Programm listen, dann werden diese Zahlen wieder in ihrer ursprünglichen Form dargestellt. Da AMOS-Basic alle Zahlen in einer genormten Art darstellt, kann es manchmal zu einer kleinen Diskrepanz zwischen der Zahl, die Sie eingegeben haben, und der Zahl, die in Ihrem Listing erscheint, kommen. Der Wert dieser Zahl bleibt jedoch völlig unverändert.

Fließkommakonstanten unterscheiden sich von den ganzzahligen Werten durch das Komma, das in der englischen Schreibweise als Dezimalpunkt dargestellt wird. Wenn das Komma - bzw der Punkt - nicht verwendet wird, dann wird die Zahl immer wie eine ganze Zahl behandelt, selbst wenn sie in einer Fließkommarechnung auftaucht. Nehmen wir einmal folgendes Beispiel:

```
For X=1 To 10000
  A#=A#+2
Next X
```

Jedes Mal, wenn dieser Ausdruck im Programm berechnet wird, wird die "2" mühsam in eine reelle Zahl umgerechnet. Das macht diese Routine erheblich langsamer als die gleichwertige Routine im folgenden Programm:

```
For X=1 To 10000
  A#=A#+2.0
Next X
```

Dieses Programm läuft über 25% schneller als das vorherige ab, weil die Konstante jetzt direkt im Fließkomma-Format gespeichert ist. Deshalb sollten Sie nie vergessen, bei Fließkomma-Zahlen das Komma bzw den Dezimalpunkt zu setzen, auch wenn die Konstante eine ganze Zahl ist. Übrigens, wenn Sie ganze und Fließkomma-Zahlen mischen, dann wird das Ergebnis immer als reelle Zahl dargestellt. Hier ist ein Beispiel:

```
Print 19.0/2
9.5
Print 3.141+10
13.141
```

Arithmetische Operationen

Die folgenden Rechenoperationen können numerisch ausgedrückt werden:

^	Potenzrechnung
/und *	Division und Multiplikation
MOD	Modulo-Operator (Rest aus einer Division)

+ und -	Plus und Minus
AND	Logisches UND
OR	Logisches ODER
XOR	Logisches XOR

Wir haben diese Operation nach ihrer Wertigkeit geordnet. Darunter versteht man die Reihenfolge, in welcher die verschiedenen Rechenschritte eines arithmetischen Ausdrucks durchgeführt werden. Die Rechnungen mit der höchsten Wertigkeit werden immer zuerst berechnet. Das sieht in der Praxis zum Beispiel so aus:

```
Print 10+2*5-8/4+5^2
```

Der Ausdruck wird in dieser Reihenfolge berechnet:

5^2 (entspricht 5*5)	= 25
2*5	= 10
8/4	= 2
10+10	= 20
20-2	= 18
18+25	= 43

Wenn Sie möchten, daß dieser Ausdruck in einer anderen Reihenfolge berechnet wird, dann müssen Sie die Rechenschritte, die zuerst ausgeführt werden sollen, in runde Klammern setzen:

```
Print (10+2)*(5-8/4+5)^2
```

Dann erhalten Sie das Ergebnis 12*(8^2) oder 12*64 oder 768. Wie Sie sehen, hat das Setzen von zwei Klammern die Berechnung grundlegend verändert.

Da wir gerade beim Thema Berechnungen sind, wollen wir auch gleich drei Anweisungen erwähnen, die Ihre Programme ganz erheblich beschleunigen können.

INC *(Erhöht eine Integervariable um eins)*

INC var

INC addiert mit einer einzigen 68000-Anweisung 1 zu einer Integervariablen. Das ist die logische Entsprechung des Ausdrucks var=var+1, aber es geht eben viel schneller. Zum Beispiel:

```
A=10:Inc A:Print A
11
```

DEC *(Erniedrigt eine Integervariable um eins)*

DEC var

Mit dieser Anweisung wird von der Integervariablen var 1 subtrahiert. Hier ein Beispiel:

```
A=2
Dec A
Print A
1
```

ADD *(Schnelle Ganzzahl-Addition)*

ADD v,exp [,base TO top]

Die normale Ausführung dieser Anweisung fügt sofort das Ergebnis der Berechnung exp zur Ganzzahl-Variablen v hinzu. Sie entspricht der Zeile: V=V+EXP

Der einzige grundlegende Unterschied zwischen den beiden Ausdrücken besteht darin, daß ADD diese Berechnung ungefähr 40% schneller durchführt. Beachten Sie bitte, daß die Variable v eine ganze Zahl sein muß. Zum Beispiel:

```
Timer=0
For X=1 To 1000
   Add T,X
Next X
Print T,Timer
500500 7
```

Die zweite Version von ADD ist etwas komplizierter. Sie entspricht effektiv dem folgenden Ausdruck:

```
V=V+A
If V<Base Then V=TOP
If V>Top Then V=BASE
```

Wie die erste Form von ADD ist auch dieser Befehl erheblich schneller als die einzelnen Anweisungen. Hier ist ein Beispiel:

```
Dim A(10)
For X=0 To 10:A(x)=X:Next X
V=0
Repeat
   Add V,1,1 To 10
```

```
    Print A(V)
Until V=100:rem Dies ist eine Endlosschleife da V immer kleiner als 10 ist!
```

Wie Sie sehen, ist ADD ideal für den Einsatz von Schleifen die im Kreis laufen oder sich in Ihren Spielen wiederholen.

Rechenoperation bei Zeichenketten (Strings)

Wie in den meisten anderen Basic-Versionen kann man auch in AMOS zwei Zeichenketten (Strings) reibungslos miteinander verbinden.

```
A$="AMOS"+"BASIC"
Print A$
AMOS Basic
```

Aber AMOS kann nicht nur addieren, sondern auch subtrahieren! Wie Sie anhand der folgenden Beispiele sehen, werden die in der 2. Zeichenkette in Anführungszeichen gesetzten Zeichen aus der 1. Zeichenkette entfernt:

```
Print "AMOS BASIC"-"S"
AMO BAIC
Print "AMOS BASIC"-"AMOS"
BASIC
Print "Eine Kette von Zeichen"-" "
EineKettevonZeichen
```

Die beiden Zeichenketten werden Zeichen für Zeichen anhand der Ascii-Werte der entsprechenden Buchstaben verglichen. Das sieht so aus:

```
"AA"<"BB"
"Dateiname"="Dateiname"
"X&">"X#"
"HALLO"<"hallo"
```

Geben Sie das folgende Programm ein:

```
Input "Gib Deinen Vornamen ein";C$
Input "Gib Deinen Nachnamen ein";S$
If C$>S$ Then Print S$;" ";C$ Else Print C$;" ";S$
```

Parameter

Alle Werte, die Sie in einer AMOS Basic-Anweisung eingeben, heißen Parameter, das sind zum Beispiel:

```
Inc N
Add A,10
Ink 1,2,3
```

Die Parameter in der obigen Anweisung sind N, A, 10, 1, 2 und 3. Manchmal können einige der Parameter eines Befehls aus einer Anweisung weggelassen werden. In diesem Fall wird den nicht verwendeten Werten automatisch durch den Default eine Zahl zugewiesen. Nehmen wir einmal das folgende Beispiel:

```
Ink 5,,
```

Diese Anweisung verändert die Farbe, in der Zeichnung bzw Text dargestellt werden (ink colour), hat jedoch keine Auswirkungen auf den Hintergrund (paper) oder die Farbe des Rahmens (outline). Beachten Sie bitte, daß die Kommas an ihrem üblichen Platz sind, obwohl die Werte selbst nicht erscheinen. AMOS ermittelt anhand dieser Kommas die Reihenfolge, in der die Parameter in die Anweisung eingegeben werden müssen. Sie können so zum Beispiel einen Wert in die Mitte eines Befehls einsetzen:

```
Ink ,3,
```

Jetzt wird die Farbe des Hintergrunds verändert, und die Farben von Zeichnung und Rahmen bleiben unberührt.

Dieses Prinzip kann auch auf viele andere AMOS Basic-Befehle angewendet werden. Vorausgesetzt, Sie vergessen nicht, die Kommas in ihrer ursprünglichen Stellung zu lassen, können Sie sich mit dieser Technik eine ganze Menge unnötiger Tipp-Arbeit bei der Erstellung Ihrer Programme sparen. Wenn ein Parameter unbedingt erforderlich ist, dann erhalten Sie die Warnung Falscher Funktionsaufruf. Es lohnt sich also, mit den verschiedenen Kombinationen etwas herumzuexperimentieren.

Zeilennummern und Sprungmarken (Labels)

In früheren Version von Basic mußte jede Programmzeile immer mit einer Zahl beginnen. Diese Zeilennummern stellten die Zieladresse für die GOTO- oder GOSUB-Befehle dar und wurden auch vom Basic Editor eingesetzt. Daran haben wir im Prinzip nichts auszusetzen, denn diese Methode wird auch bei STOS Basic eingesetzt, aber bei AMOS ist es eben nicht unbedingt erforderlich. Hier wird Ihnen das Numerieren der Zeilen völlig freigestellt, die Zeilennummer wird nur angegeben, damit die Kompatibilität zu STOS Basic gewährleistet ist. Jetzt fragen Sie sich sicher, wie Sie GOTO oder GOSUB ohne Zeilennummern einsetzen können. Nun, Sie ersetzen sie einfach durch Sprungmarken.

Sprungmarken

Sprungmarken eignen ausgezeichnet zur Markierung eines Punktes in Ihren AMOS-Basic Programmen. Sie bestehen aus einer Zeichenkette, die nach den gleichen Regeln wie die AMOS-Variablen erstellt wird. Man sollte die Sprungmarken immer an den

Anfang einer Zeile stellen, und man muß sie stets mit einem ":" Doppelpunkt abschließen. Zwischen der Sprungmarke und dem Doppelpunkt darf kein Leerschritt geschaltet werden. Sonst wird die Sprungmarke wie eine Prozedur behandelt und Sie erhalten die Fehlermeldung Nicht definierte Procedure. Hier ist ein einfaches Beispiel:

```
TESTLABEL: Rem Das ist eine Sprungmarke
Print "Na, Du"
Goto TESTLABEL
```

Das Programm schreibt den Ausdruck "Na, Du" wiederholt auf den Bildschirm. Das ganze kann mit Ctrl+C abgebrochen werden.

Sprungmarken sind viel leichter zu lesen als Zeilennummern. Deshalb sollten Sie sie in Ihren AMOS-Basic Programmen ausgiebig verwenden.

Prozeduren

Wenn Sie schon einmal versucht haben, ein ziemlich großes Basic-Programm zu schreiben, dann können Sie sich bestimmt gut vorstellen, wie leicht man da irgendwo in der Mitte total den Durchblick verlieren kann. Deshalb teilen heutzutage die meisten Profi-Programmierer ihre Programme in kleinere Module auf, die man Prozeduren nennt.

Durch diese Aufteilung in Prozeduren können Sie sich in Ruhe auf die Lösung eines Problems konzentrieren, ohne sich vom Rest des Programmes ablenken zu lassen. Wenn Sie dann alle Prozeduren geschrieben haben, geht das Zusammenschmieden zu einem einzigen Programm ganz schnell.

Programme, die auf Abläufen aufbauen, sind leicht zu schreiben, leicht zu überarbeiten und man kann auch die Bugs leichter finden und beseitigen. Die AMOS-Basic Prozeduren sind völlig selbständige Programm-Module, die ihre eigenen Programmzeilen, Variablen, und sogar Datenvereinbarungen haben können. Also gibts überhaupt keine Ausrede dafür, sie nicht in Ihren Programmen voll einzusetzen.

PROCEDURE *(Erstellt eine AMOS-Basic Prozedur)*

```
Procedure NAME[Parameter-Liste]
:            :
End Proc[Ausdruck]
```

So wird eine Prozedur in AMOS-Basic mit der Bezeichnung *NAME* definiert. *NAME* ist dabei eine Zeichenkette, durch die diese Prozedur identifiziert wird, sie wird genau wie eine normale Basic-Variable erstellt. Dabei können Prozeduren, Variablen und Sprungmarken durchaus übereinstimmende Namen haben. Aus dem Zusammenhang in der Zeile ermittelt AMOS automatisch, worauf Sie sich beziehen.

Die Prozeduren ähneln den GOSUB-Befehlen, die in früheren Basic-Versionen auftauchten. Es folgt ein Beispiel für eine einfache AMOS-Prozedur:

```
Procedure ANTWORT
  Print "Zweiundvierzig!"
End Proc
```

Sehen Sie, wie die Prozedur durch die Anweisung End Proc abgeschlossen wird. Dabei beachten Sie bitte, daß die Anweisungen "Procedure" und "End Proc" jeweils in einer neuen Zeile stehen müssen, sonst läuft's nicht!

Wenn Sie die Prozedur genau wie oben eingeben, und dann versuchen, sie ablaufen zu lassen, dann passiert erst einmal garnichts. Das kommt daher, weil Sie diese neue Prozedur nicht aus Ihrem Basic-Programm abgerufen haben. Und wenn Sie nun ihren Namen an eine passende Stelle in das Programm einfügen, können Sie das nachholen. Geben Sie zum Beispiel die folgende Zeile am Anfang des Programms ein, lassen Sie es ablaufen, und sehen Sie, was passiert.

```
ANTWORT
```

Achtung: Wenn Sie mehrere Prozeduren in derselben Zeile verwenden, dann sollten Sie am Ende jeder Anweisung zusätzliche Leerschritte schalten. Damit können Sie vermeiden, daß die Prozeduren mit Sprungmarken verwechselt werden. Hier ein Beispiel:

```
TEST : TEST : TEST:Rem Führt die Test-Prozedur dreimal durch
TEST:TEST:TEST:Rem Definiert die Sprungmarke TEST und führt TEST nur zweimal durch
```

Sie sehen, daß im ersten Fall wird die Test-Prozedur dreimal ausgeführt wird, im zweiten Fall nur zweimal durchgeführt und die Sprungmarke TEST definiert wird.

Alternativ dazu können Sie Ihre Prozedur auch mit einer Proc-Anweisung folgendermaßen einleiten:

```
Proc ANTWORT
```

Beispiel:

```
Rem Zeigt Ihnen genau, daß eine Prozedur aufgerufen wird,
Rem nicht nur ein Befehl.
Proc ANTWORT
Rem Das Gleiche kann auch ohne Proc erreicht werden
ANTWORT
Procedure ANTWORT
  Print "Zweiundvierzig!"
End Proc
```

Wenn Sie jetzt das Programm nochmal ablaufen lassen, dann wird die Prozedur eingesetzt, und die Antwort erscheint auf dem Bildschirm. Die Prozedur-Definition

befindet sich zwar am Ende des Programms, aber man kann Sie wirklich an jede beliebige Stelle setzen. Wenn AMOS auf eine Prozedur-Anweisung stößt, wird die Prozedur automatisch installiert, und AMOS springt zu End Proc. Das bedeutet, daß keine Gefahr besteht, daß Ihre Prozedur versehentlich abgerufen wird. Wenn Sie eine Prozedur einmal erstellt haben und nach der Überprüfung mit Ihr ganz zufrieden sind, dann können Sie sie mit der Falten-Option aus dem Hauptmenü unterdrücken.

Mit dieser Option können Sie Ihre Listings viel klarer und übersichtlicher gestalten und umfangreiche Programme auf Bugs untersuchen, ohne durch unwesentliche Einzelheiten aufgehalten zu werden. Sie können die vollständige Liste jederzeit durch den Befehl Falten wiederherstellen.

Lokale und globale Variablen

Alle Variablen, die Sie in Ihren Prozeduren definieren, hängen wiederum von den anderen Variablen in Ihrem Programm ab. Diese Variablen werden als "lokal" bezeichnet, denn sie beziehen sich nur auf Ihre spezielle Prozedur. Wir wollen das an folgendem Beispiel verdeutlichen:

```
A=1000:B=42
TEST
Print A,B
Procedure TEST
   Print A,B
End Proc
```

Daraus wird deutlich, daß A und B zwei völlig unterschiedliche Variablen bezeichnet, je nachdem, ob Sie innerhalb oder außerhalb der Prozedur TEST eingesetzt werden. Die Variablen, die außerhalb einer Prozedur auftreten, sind global und nicht aus einer Prozedur heraus zugänglich. Nehmen wir ein anderes Beispiel:

```
Dim A(100)
For V=1 to 100: A(V)=V:Next V
TEST_FLAG=1
APRINT
End
Procedure APRINT
   If TEST_FLAG=1
      For P=1 to 100
         Print A(P)
      Next P
   Endif
Endproc
```

Dieses Programm sieht vielleicht ganz harmlos aus, aber es enthält zwei böse Fehler. Erstens wird TEST_FLAG innerhalb der Prozedur immer den Wert Null haben und die Schleife zwischen dem IF und dem ENDIF wird nie ausgeführt. Das ist kommt daher,

daß die Version von TEST_FLAG innerhalb der Prozedur von der im Hauptprogramm definierten Version völlig unabhängig ist. Es wird ihr nämlich der Wert Null zugewiesen, ebenso wie allen anderen Variablen, wenn sie das erste Mal eingesetzt werden.
Zweitens läuft das Programm überhaupt nicht! Da das Global-Array A() außerhalb von APRINT definiert wurde, quittiert AMOS-Basic die Zeile:

```
Print A(P)
```

sofort mit der Fehlermeldung *Feld nicht dimensioniert.* Solch einen Fehler kann man schrecklich leicht machen, und deshalb sollten Sie die Prozeduren unbedingt wie unabhängige Programme mit eigenen Variablen und Anweisungen behandeln. Übrigens, kommen Sie bloß nicht auf die Idee, die gleichen Namen für die Variablen innerhalb und außerhalb einer Prozedur zu verwenden. Dann glauben Sie nämlich vielleicht, daß es sich dabei um dieselben Variablen handelt, und schon schnappt die Falle zu! Und dann wissen Sie nicht, wie es plötzlich zu diesen unerklärlichen Fehlern in Ihrem Programm kommt.

Glücklicherweise gibt es ein paar Erweiterungen in diesem System, durch die Sie ganz leicht Information zwischen einer Prozedur und Ihrem Hauptprogramm übertragen können. Wenn Sie erst mit diesen Befehlen vertraut sind, dann fällt es Ihnen nicht mehr schwer, die Prozeduren erfolgreich in Ihren Programmen einzusetzen.

Parameter und Prozeduren

Sie haben die Möglichkeit, eine Liste von "Parameter-Definitionen" in Ihre Prozeduren einzubauen. So entsteht eine Gruppe von lokalen Variablen, die dann direkt aus dem Hauptprogramm geladen werden kann. Hier ist ein Beispiel:

```
Procedure HALLO[NAME$]
   Print "Hallo";NAME$
End Proc
```

Der Wert, der in NAME$ geladen werden soll, wird als Teil des Prozedur-Abrufs in die eckigen Klammern eingegeben. So könnte die Prozedur HALLO folgendermaßen ausgeführt werden:

```
Rem Lädt N$ in NAMENS$ und steigt in Prozedur ein
Input "Wie heißt Du denn";n$
HALLO[n$]
Rem Lädt die wörtliche Zeichenkette "Stephen" in NAME$ und ruft HALLO auf
HALLO["Stephen"]
```

Wie Sie sehen, kann das Parameter-System sehr vielseitig eingesetzt werden und funktioniert mit Variablen ebenso gut wie mit Konstanten. Das einzig Wichtige dabei ist die Art der Variablen. Mit dieser Methode können ganze oder reelle Zahlen sowie Zeichenketten eingegeben werden. Sie können diese Funktion jedoch nicht auf ganze

Arrays anwenden. Wenn Sie mehrere Parameter eingeben möchten, dann sollten Sie die einzelnen Variablen durch Kommas voneinander trennen. Zum Beispiel so:

```
Procedure POWER[A,B]
Procedure MERGE[A$,B$,C$]
```

Diese Prozeduren können mit Zeilen wie:

```
POWER[10,3]
MERGE["Eins","Zwei","Drei"]
```

abgerufen werden.

Gemeinsame Variablen (Shared)

Man kann die Daten zwischen einer Prozedur und dem Hauptprogramm auch durch die Anweisung SHARED übertragen.

SHARED *(Definiert eine Liste globaler Variablen)*

SHARED Variablen-Liste

Die Anweisung SHARED wird in eine Prozedur-Definition eingesetzt und nimmt eine Liste von AMOS-Basic Variablen auf, die durch Kommas voneinander abgetrennt sind. Diese Variablen werden jetzt wie globale Variablen behandelt und sind direkt aus dem Hauptprogramm zugänglich. Alle so definierten Arrays sollten Sie natürlich zuvor in Ihrem Hauptprogramm dimensioniert haben. Zum Beispiel:

```
A=1000:B=42
TEST
Print A,B
Procedure Test
   Shared A,B
    A=A+B:B=B+10
End Proc
```

TEST kann jetzt die Angaben zu den globalen Variablen A und B lesen und darstellen. Wenn Sie ein Array zur gemeinsamen Nutzung zugänglich machen möchten, dann sollten Sie es folgendermaßen definieren:

```
Shared A(),B#(),C$():rem Arrays A,B# und C$ werden gemeinsam genutzt
```

GLOBAL *(Definiert eine Liste von globalen Variablen aus dem Hauptprogramm)*

GLOBAL Variablen-Liste

Wenn Sie ein großes Programm schreiben, dann kommt es oft vor, daß eine Reihe von Prozeduren dieselbe Gruppe von globalen Variablen teilen. Das ist eine einfache Methode für die Übertragung von großen Mengen an Information zwischen Ihren verschiedenen Prozeduren. Um diesen Prozeß zu vereinfachen, haben wir einen einzigen Befehl eingebaut, der direkt in Ihrem Hauptprogramm eingesetzt werden kann. GLOBAL definiert eine Liste von Variablen, die überall in Ihrem Basic- Programm zugänglich sind, ohne daß Sie die Parameter in Ihren Prozedur- Definitionen einzelnen mit SHARED kennzeichnen müssen. Das sieht zum Beispiel so aus:

```
A=1000:B=42
Global A,B
TEST1
Print A,B
TEST2
Print A,B
Procedure TEST1
  A=A+B:B=B+10
End Proc
Procedure TEST2
  A=A*B:B=B+10
End Proc
```

Ausgeben von Werten durch eine Prozedur

Wenn eine Prozedur einen Wert ausgegeben soll, der ausschließlich lokal ist, dann muß die folgende Anweisung eingesetzt werden. So wird die Information darüber, wo die lokale Variable zu finden ist, auf den Befehl, mit dem man die Prozedur aufruft, übertragen.

PARAM *(Ausweisen eines Parameters aus einer Prozedur)*

PARAM

Die PARAM-Funktionen bieten Ihnen eine einfache Möglichkeit, das Ergebnis einer Prozedur auszugeben. Sie entnehmen das Ergebnis eines beliebigen Ausdrucks der End Proc Anweisung und geben es in einer der Variablen PARAM, PARAM# oder PARAM$ - je nach Art der Variable - wieder aus. Beispiel:

```
MERGE_STRINGS["Amos","","Basic"]
Print PARAM$
```

```
Procedure MERGE_STRINGS[A$,B$,C$]
   Print A$,B$,C$
End Proc[A$+B$+C$]
```

Beachten Sie hier bitte, daß Sie auf diese Weise durch End Proc nur einen einzigen Parameter erhalten. Die PARAM-Funktionen enthalten immer das Ergebnis der zuletzt ausgeführten Prozedur. Hier ist ein anderes Beispiel, das diesmal den Einsatz der PARAM# Funktion demonstriert.

```
WÜRFEL[3.0]
Print Param#
Procedure WÜRFEL[A#]
   W#=WÜRFEL#*WÜRFEL#*WÜRFEL#
End Proc [W#]
```

Aus einer Prozedur aussteigen

POP PROC *(Sofort aus einer Prozedur aussteigen)*

POP PROC

Normalerweise kehrt die Prozedur erst in das Hauptprogramm zurück, nachdem es den Befehl End Proc erreicht hat. Manchmal müssen Sie aber ganz schnell vorher aus einer Prozedur aussteigen, und dann können Sie sie mit dem Befehl POP PROC abbrechen. Dafür ein Beispiel:

```
Procedure BEENDEN
   For LANGWEILIG=1 to 10000
        If LANGWEILIG=10 Then Pop Proc
   NEXT LANGWEILIG
   Print "Diese Zeile wird nie ausgeführt"
End Proc
```

Lokale Daten-Definitionen

Die Daten-Definitionen, die in einer Ihrer Prozeduren festgelegt sind, werden von den im Hauptprogramm definierten völlig getrennt verwahrt. Das bedeutet, daß jede Prozedur über ihren eigenen Datenbereich verfügen kann. Dazu ein Beispiel:

```
Read A$:Print A$,
BEISPIEL
Read B$:Print B$,
Procedure BEISPIEL
   Read X$,Y$
```

```
    Print X$,Y$
    Data "Basic", "ist"
End Proc
Data "AMOS","ganz irre!"
```

Tips und Tricks

Hier ein paar Tricks, damit Sie Ihre AMOS-Basic Prozeduren wirklich voll ausnutzen können:

- Eine Prozedur kann sich durchaus selbst aufrufen, aber diese Möglichkeit ist durch den verfügbaren Speicherplatz für die lokalen Variablen eingeschränkt. Wenn der Speicher für Ihr Programm zu klein wird, dann erhalten Sie die entsprechende Fehlermeldung.

- Alle lokalen Variablen werden automatisch auf Null gesetzt, nachdem die Prozedur ausgeführt wurde.

```
Procedure ADDIEREN
    A=A+1 : Print A
End Proc
```

  Ganz egal, wie oft Sie diese Prozedur aufrufen, es wird immer derselbe Wert (1) angezeigt.

- AMOS Prozeduren gleichen den Subroutinen, die man mit den SUB-Befehlen in Amiga Basic erstellen kann. Der einzig nennenswerte Unterschied zwischen ihnen ist, daß man Arrays nicht als Parameter übertragen kann. Wenn Sie aus einer Prozedur heraus Zugang zu einem Array haben wollen, dann müssen Sie es stattdessen mit SHARED kennzeichnen.

Speicherbanken

AMOS-Basic enthält eine Reihe von leistungsstarken Eigenschaften für den Einsatz von Sprites, BOBs und Musik. Die von diesen Funktionen benötigten Daten müssen zusammen mit dem Basic-Programm gespeichert werden. AMOS-Basic verwendet einen besonderen Satz von 15 Speicherbereichen für diesen Zweck, die sogenannten Speicherbanken.

Jede Bank wird mit einer eigenen Nummer zwischen 1 und 15 bezeichnet. Viele dieser Banken können für alle möglichen Arten von Daten verwendet werden, aber manche sind ausschließlich einer bestimmten Art von Informationen, wie zum Beispiel Sprite-Definitionen, vorbehalten. Alle Sprite-Bilder sind in Bank 1 gespeichert. Sie können zum Beispiel mit einer solchen Anweisung in den Speicher geladen werden:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
```

Es gibt zwei verschiedene Arten von Speicherbanken, permanente und temporäre. Die permanenten Speicherbanken muß man nur einmal definieren, und sie werden dann immer automatisch mit Ihrem Programm gespeichert. Die temporären Speicherbanken sind viel empfindlicher und müssen jedesmal, wenn ein Programm abläuft, neu initialisiert werden. Im Gegensatz zu den permanenten Speicherbanken können die temporären mit dem CLEAR-Befehl aus dem Speicher gelöscht werden.

Die Arten von Speicherbanken

AMOS-Basic unterstützt die folgenden Arten von Speicherbanken:

Art	Speichert	Beschränkungen	Typ
Sprites	Definition von Spritesoder BOBs	Nur Bank 1	Permanent
Icon	Enthält sämtliche Icon-Definitionen	Nur Bank 2	Permanent
Musik	Enthält die Daten des Sound Tracks	Nur Bank 3	Permanent
Amal	Wird für Amal Daten verwendet	Nur Bank 4	Permanent
Tracker	Speichert unkonvertierte Soundtrackermusiken	Bank 1-15	Permanent
Muster	Die Muster-Daten	Bank 1-15	Permanent
Menü	Speichert Menü-Definition	Bank 1-15	Permanent
Chip Work	Temporärer Arbeitsspeicher	Bank 1-15	Temporär
Chip Data	Permanenter Arbeitsspeicher	Bank 1-15	Permanent
Fast Work	Temporärer Arbeitsspeicher	Bank 1-15	Temporär
Fast Data	Permanenter Arbeitsspeicher	Bank 1-15	Permanent

RESERVE *(Bank reservieren)*

RESERVE AS Typ, Bank, Länge

Die Speicherbanken für die Sprites oder BOBs werden von AMOS automatisch zugeordnet. Der Befehl RESERVE ermöglicht es Ihnen, zusätzliche Speicherbanken, die Sie vielleicht benötigen, zu reservieren. Es gibt für jede der verschiedenen Bankarten eine besondere Ausführung der RESERVE- Anweisung.

RESERVE AS WORK Banknummer, Länge

Reserviert Länge: Anzahl der Bytes, die als temporärer Arbeitsspeicher zur Verfügung gestellt werden. Wenn möglich, wird dieser Speicher als Fast-Speicher (Zwischenspeicher) zugewiesen, deshalb sollten Sie diesen Befehl nicht in Verbindung mit Anweisungen einsetzen, die Zugang zum Blitter-Chip des Amiga erfordern.

RESERVE AS CHIP WORK Banknummer, Länge

Weist einen Arbeitsspeicher der angegebenen Länge unter der Verwendung von Chip

RAM zu. Sie können mit der CHIP FREE Funktion überprüfen, ob ausreichend Chip RAM verfügbar ist.

RESERVE AS DATA Banknummer, Länge

Reserviert eine permanente Speicherbank mit der entsprechenden Byte-Länge. Dieser Datenbereich wird wenn möglich dem Fast-Speicher zugeordnet.

RESERVE AS CHIP DATA Banknummer, Länge

Reserviert Bytes der angegebenen Länge im Chip RAM-Speicher. Diese Speicherbank wird automatisch mit Ihrem AMOS-Programm gespeichert.

Die Banknummer kann jede der Nummern von 1 bis 15 sein, es ist aber ratsam, die Banken 1 bis 5 dabei in Ruhe zu lassen, da sie normalerweise vom System reserviert sind. Beachten Sie, daß die einzige Beschränkung der Banklänge durch den verfügbaren Speicherplatz gegeben ist.

LISTBANK *(Listet die genutzten Banken auf)*

Der Befehl LISTBANK bewirkt eine Auflistung der Banken, die derzeitig von Programm belegt werden, zusammen mit ihrer Adresse und Größe. Die Liste besitzt das folgende Format:

Nummer	Typ	Anfang	Länge
1	-Sprites	S:$040F60	L:$00002F
2	-Work	S:$05F7A0	L:$014000

S:=Die Adresse des Anfangs der Speicherbank in Hexadezimal-Zahlen.
L:=Die Länge der Speicherbank in Hexadezimal-Zahlen.

Normalerweise wird die Länge der Speicherbank in Bytes angegeben, aber im Fall von Sprite und Icon stellt der Wert stattdessen die Gesamtanzahl der Bilder in der Bank dar. Der Grund dafür ist, daß das Bild an einer beliebigen Stelle im Speicher des Amiga sein kann, und die Bank infolgedessen kein durchgängiger Speicherblock ist. Deshalb verwenden Sie bitte nicht BSAVE bei einer Sprite-Bank, sondern einfach nur SAVE "Dateiname.ABK".

Das Löschen einer Speicherbank

Im Laufe eines Programmes kann es vorkommen, daß Sie einige der Speicherbanken aus dem Speicher löschen müssen, damit Sie zusätzliche Daten laden können. Es kann sein, daß Sprites für einen neuen Teil des Spiels verändert werden müssen, oder daß Sie ein bestimmtes Musik-Stück einbauen möchten. Mit dem ERASE-Befehl können Sie die Daten schnell löschen.

ERASE *(Löschen einer Speicherbank)*

ERASE b

ERASE löscht den Inhalt einer Speicherbank. Die Banknummer b kann im Bereich von 1 bis 15 liegen. Der durch das Löschen der Bank freigewordene Speicherplatz steht dann für Ihr Programm zur Verfügung.

Parameter-Funktionen der Banken

Wenn Sie direkten Zugang zu den Daten in der Speicherbank über Befehle wie "Poke", "Doke" und "Loke" haben möchten, dann wenden Sie die folgenden Befehle an, um die Speicheradresse und die Größe dieser Bank zu finden.

=START *(Für die Startadresse der Speicherbank)*

s=START(b)

Diese Funktion ermittelt die Startadresse der Speicherbank Nummer b. Nachdem sie einmal reserviert ist, ändert sich die Adresse einer Speicherbank nicht mehr. Deshalb wird das Ergebnis, das diese Funktion ausweist, für die gesamte Lebensdauer dieser Bank gelten. Hier ein Beispiel:

```
Reserve As Work 3,2000
Print Start(3)
```

=LENGTH *(Ermittelt die Länge der Speicherbank)*

l=LENGTH(b)

Die Funktion LENGTH gibt die Länge der Speicherbank Nummer b in Bytes an. Wenn die Speicherbank Sprites oder Icons enthält, dann wird stattdessen deren Anzahl angegeben. Der Wert "Null" zeigt an, daß die Bank nicht existiert. Zum Beispiel:

```
Reserve As Work 6,1000
Print Length(6)
Erase 6
Print Length(6)
```

Das Laden und Speichern von Banken

Manche Programme erfordern ein ganze Reihe von Informations-Banken, ein gutes Beispiel dafür ist sind Adventures. Da sind eine ganze Menge Grafik und Sound- Effekte

für die verschiedenen Szenarios erforderlich. Ein Amiga 500 hätte ganz schön zu kämpfen, wenn er alle Daten gleichzeitig parat haben müßte, und deshalb ist es am sinnvollsten, die Daten dann zu laden, wenn sie benötigt werden.

LOAD *(Laden von einer oder mehreren Banken)*

LOAD "Dateiname"[,n]

Welche Wirkung dieser Befehl hat, hängt von der Art der Datei, die Sie laden, ab. Wenn die Datei mehrere Banken umfaßt, dann werden alle derzeitig geladenen Speicherbanken gelöscht, bevor die neuen Banken von der Diskette geladen werden. Wenn Sie aber nur eine Bank laden, dann wird nur diese Bank ersetzt. Wahlweise kann auch festgelegt werden, in welche Bank Ihre Daten geladen werden sollen. Wird nicht extra ein Ziel eingegeben, so werden die Daten in die Bank geladen, in der sie ursprünglich gespeichert waren.

Bei Sprite-Banken verläuft das Ganze etwas anders. In diesem Fall schaltet der Parameter n zwischen zwei unterschiedlichen Lade-Vorgängen hin und her. Wenn n nicht eingegeben wird oder den Wert Null hat, dann überschreiben die neuen Sprites aktuelle Bank völlig. Jeder andere Wert für n veranlaßt, daß die neuen Sprites an diese Bank angehängt werden. So können Sie mehrere Sprite-Dateien in einem Programm kombinieren. Hier ein Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
```

SAVE *(Speichert eine oder mehrere Banken auf Diskette)*

SAVE "Dateiname"[,n]

Mit diesem Befehl speichern Sie Ihr Speicherbank(en) auf Diskette. Es gibt zwei mögliche Formate:

```
SAVE "Dateiname.ABK"
```

So werden alle zur Zeit definierten Banken in eine einzige Datei auf der Diskette gespeichert.

```
SAVE "Dateiname.ABK",n
```

Diese erweiterte Form des Befehls speichert nur die Speicherbank mit der Nummer n. Verwenden Sie beim Speichern immer die Erweiterung .ABK, denn sie zeigt an, daß diese Datei eine oder mehrere Speicherbanken enthält.

BSAVE *(Speichert einen nicht formatierten Speicherblock in binärem Format)*

BSAVE Datei$, Start TO Ende

Der Speicherinhalt zwischen Start und Ende wird auf der Diskette in der Datei$ gespeichert. Diese Daten werden ohne besondere Formatierung gespeichert. Zum Beispiel:

```
Bsave "Test",Start(7) TO Start(7)+Length(7):Rem Speichert Speicherbank
```

In diesem Beispiel werden die Daten in Speicherbank 7 auf Diskette gespeichert.

Der Unterschied zwischen dieser Datei und einer Datei, die als normale Speicherbank mit SAVE gespeichert wurde, besteht im wesentlichen darin, daß SAVE eine spezielle Kopfzeile mit Information über die Speicherbank schreibt. Da diese Kopfzeile fehlt, wenn die Bank mit BSAVE gespeichert wird, kann sie auch nicht mit LOAD wieder geladen werden.

Achtung: Die Sprite- und Icon-Banken werden nicht als Speicherblock behandelt. Jedes einzelne Bild kann irgendwo im Speicher sitzen. Und weil AMOS dieses flexible System der Datenspeicherung einsetzt, können diese Speicherbanken nicht mit BSAVE gespeichert werden.

BLOAD *(Lädt binäre Information in eine angegebene Speicheradresse oder Bank)*

BLOAD Datei$,Adr

Der Befehl BLOAD lädt eine Datei mit binären Daten in den Speicher. Die empfangene Information wird in keiner Weise verändert. Auch für diese Funktion gibt es zwei Ausführungsmöglichkeiten:

```
Bload Datei$,Adr
```

Die Datei Datei$ wird von der Diskette in den Speicher an der genannten Adresse geladen.

```
Bload Datei$,bank
```

Die benannte Datei wird in die angegebene Bank geladen. Diese Bank muß vorher reserviert worden sein, sonst erhalten Sie eine Fehlermeldung. Sie vergewissern sich besser vorher, daß die Datei die Kapazität der reservierten Bank nicht übersteigt, sonst laufen die Daten über und zerstören andere Speicherbereiche.

Vertauschen der Banken

BANK SWAP *(Flippt zwei Banken im Speicher)*

BANK SWAP Nummer1,Nummer2

Durch diese Anweisung werden die Zeiger auf zwei Banken vertauscht. Das ist sehr nützlich, wenn Sie eine Icon-Bank in eine Sprite-Bank verwandeln möchten. Dazu folgendes Beispiel:

```
Bank Swap 1,2
```

Oder wenn Sie mehr als eine Musikbank gleichzeitig haben möchten, zum Beispiel:

```
Bank Swap 3,5
```

usw.

Speicherfragmentierung

Manchmal, wenn Sie fleißig gearbeitet haben, kann es passieren, daß Sie die Meldung "Kein Speicherplatz mehr" erhalten, obwohl Sie der Status-Zeile entnehmen können, daß noch genug Speicher vorhanden ist. Das ist kein AMOS-Bug, sondern eine unvermeidbare Nebenwirkung des Systems, nach dem der Amiga den Speicher zuweist.

Sie müssen sich den Amiga-Speicher wie einen Kuchen vorstellen. Wenn Sie den Kuchen einmal angeschnitten haben, können Sie das Stück nie wieder genau an der ursprünglichen Stelle einfügen. Jedes Mal, wenn Sie Speicher reservieren, wird ein Stück von dem Speicherkuchen abgeschnitten. Und wenn Sie dann dem System diesen Speicher wieder zurückgeben, dann wird er in eine besondere "Schicht" von gegenwärtig ungenutzten Speichersegmenten geschoben. Vielleicht können Sie sich den Speicher als "Schichttorte" vorstellen!

Wenn Sie dann das nächste Mal einen Speicherbereich reservieren, sieht AMOS im ungenutzten Speicher nach, ob da ein Stück der erforderlichen Größe dabei ist. Ist ein Stück größer als erforderlich, dann schneidet AMOS es entzwei und behält das nicht benötigte Stück. Nach einer Weile besteht der gesamte Speicher aus ganz kleinen Stücken, und wenn Sie dann mehr Speicher verlangen, erhalten Sie die Fehlermeldung "Kein Speicherplatz mehr", weil kein Stück für Ihren Bedarf groß genug ist.

Es gibt nur einen Weg, dieses Problem zu lösen. Sie müssen den Amiga ausschalten und neu starten. Dadurch kehrt der Speicher in seinen Originalzustand zurück, und Sie können ihn wieder so zuweisen, wie Sie ihn brauchen.

Übrigens taucht dieses Problem nur während der Programmentwicklung auf. Wenn Sie den gesamten Speicherplatz am Anfang Ihrer Programme reservieren, dann werden Sie sich bei keinem Ihrer Programme mit diesem Problem herumschlagen müssen.

Wohin mit den Variablen?

Alle Variablen werden als Default in einen Speicherbereich mit einer Länge von genau 8 KB gespeichert. Obwohl das ziemlich mager scheint, kann man darin ungefähr zwei Seiten normalen Text oder 2000 Zahlen unterbringen. Wir haben diesen Speicher extra so knapp bemessen, damit wir soviel Platz wie möglich für Ihre Schirme und Speicherbanken zur Verfügung haben.

Nur keine Panik! Dieser Bereich kann direkt aus Ihren Basic-Programmen heraus mit dem einfachen Befehl SET BUFFER vergrößert werden. Damit liegt die einzige Beschränkung des verfügbaren Speichers für Ihre Arrays und Zeichenketten im Gesamtspeicher Ihres Computers.

SET BUFFER *(Legt die Größe des Speicherbereichs für die Variablen fest)*

SET BUFFER n

Legt die Größe des Speicherbereichs, der für die Variablen zur Verfügung steht, auf n KB fest. Dies muß die ERSTE Anweisung in Ihrem Programm sein (einschließlich Rems). Sonst erhalten Sie die entsprechende Fehlermeldung. Ein Beispiel für diese Funktion finden Sie in **BEISPIEL 4.1** im MANUAL-Unterverzeichnis.

SET BUFFER sollten Sie immer einsetzen, wenn Sie die Fehlermeldung Kein Zeichenkettenspeicher mehr erhalten. Vergrößern Sie den Speicher in Schritten von 5 KB, bis der Fehler verschwindet. Wenn Ihnen dabei der Speicher ausgeht, dann werden Sie wohl versuchen müssen, die Anforderungen Ihres Programmes etwas zurückzuschrauben. Tips dazu finden Sie bei den Befehlen CLOSE WORKBENCH und CLOSE EDITOR.

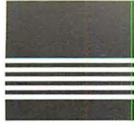
=FREE *(Gibt den freien Speicher für den Variablen-Bereich an)*

f=FREE

Mit dem Befehl FREE können Sie die Anzahl der Bytes abrufen, die gegenwärtig für das Speichern Ihrer Variablen verfügbar sind. Dieser Wert kann nach Bedarf mit dem obengenannten SET BUFFER Befehl erhöht werden.

Wenn Sie FREE abrufen, dann wird der Bereich für die Variablen so umstrukturiert, daß soviel Speicher wie möglich für Ihre Variablen frei wird. Das nennt man auch die "Müllabfuhr" und normalerweise wird es ohnehin automatisch durchgeführt.

Weil AMOS-Basic so leistungsstark ist, geht die ganze Geschichte praktisch blitzschnell. Aber wenn Ihr Bereich für die Variablen sehr groß ist, und Sie viele Zeichenketten verwendet haben, dann kann es schon ein paar Sekunden dauern. Das kann natürlich zu einer unerwarteten Verzögerung beim Ablauf Ihrer Programme führen. Da die Müllabfuhr aber unumgänglich ist - wie im richtigen Leben ja auch - sollten Sie den Aufruf des FREE-Befehls vielleicht dort einbauen, wo er in Ihren Programmen den geringsten Schaden anrichten kann.



5: String-Funktionen

AMOS-Basic enthält ein komplettes Sortiment an Anweisungen zur Handhabung von Zeichenketten (Strings). Dabei haben wir uns besonders bemüht, die normale Basic-Syntax zu verwenden, und wenn Sie schon ein erfahrener Basic- Programmierer sind, dann kennen Sie die meisten dieser Befehle bereits.

=LEFT$= *(Gibt die Zeichen ganz links an)*

d$=LEFT$(s$,n)
LEFT$(d$,n)=s$

Der Befehl LEFT$ liest die ersten n Zeichen der Zeichenkette s$ ganz links und kopiert sie in die Zielfolge d$.

Wie Sie sehen, gibt es im allgemeinen zwei Versionen dieses Befehls. Die erste ist eine Funktion, die eine neue Zielfolge d$ aus den ersten n Zeichen der Quelle s$ erstellt. Zum Beispiel:

```
Print Left$("AMOS Basic",4)
AMOS

A$=Left$("0123456789ABCDEF",10)
Print A$
0123456789

Do
   Input "Kette eingeben?";S$
   Input "Eine Kette von Zeichen";N
   Print Left$(S$,N)
Loop
```

Bei der zweiten Version dieses Befehls werden die n Zeichen ganz links in der Zielfolge durch die entsprechenden Zeichen der Quelle s$ ersetzt. Zum Beispiel:

```
A$="****Basic"
Left$(A$,4)="AMOS"
Print A$
AMOS Basic
```

=RIGHT$= *(Gibt die Zeichen ganz rechts an)*

d$=RIGHT$(s$,n)
RIGHT$=(d$,n)=s$

Mit RIGHT$ werden die ersten n Zeichen der Kette s$ ganz rechts in die Zeichenkette d$ kopiert. Beispiele:

```
Print Right$("AMOS Basic",5)
Basic
A$=Right$("0123456789ABCDEF",10)
Print A$
6789ABCDEF

Do
   Input "Kette eingeben?",V$
   Input "Anzahl der Zeichen eingeben?";N
   Print Right$(V$,N)
Loop
```

Wie bei LEFT$ gibt es auch von RIGHT$ eine zweite Version, die wie eine Basic-Anweisung aufgebaut ist.

RIGHT$(d$,n)=s$

Damit werden die ersten n Zeichnen der Quellenfolge s$ ganz rechts in die Zielfolge d$ geladen. Alle übrigen Zeichen in s$ werden völlig ignoriert. Beispiel:

```
A$="AMOS*****"
Right$(A$,5)="Basic"
Print A$
AMOS Basic
```

=MID$= *(Gibt Zeichen aus der Mitte einer Zeichenkette an)*

d$=MID$(s$,p,n)
MID$(d$,p,n)=s$

Durch die MID$-Funktion erhält man die Mitte der Zeichenkette s$. p ist dabei der Anfang der Zeichen, und n gibt die Anzahl der gewünschten Zeichen an. Wenn für n kein Wert angegeben wird, dann werden alle Zeichen bis zum Ende der Kette erfaßt. Hier einige Beispiele:

```
Print Mid$("AMOS Basic",6)
Basic
Print Mid$("AMOS Basic",6,3)
Bas
```

```
Do
  Input "Kette eingeben";V$
  Input "Ausgangsposition und Anzahl der Zeichen eingeben";S,N
  Print Mid$(V$,S,N)
Loop
```

Es gibt auch folgende MID$-Anweisung:

MID$(d$,p,n)=s$

Diese Version von MID$ lädt n Zeichen in d$, angefangen von Position p+1 in s$. Wenn für n kein Wert angegeben ist, dann werden alle Zeichen bis zum Ende der Quellenfolge s$ ersetzt. Beispiele:

```
A$="AMOS*****"
Mid$(A$,5)="Magic"
Print A$
AMOS Magic
Mid$(A$,5,3)="Bas"
Print A$
AMOS Basic
```

```
Do
  Input "Zielkette eingeben";V$
  Input "Subkette eingeben";T$
  Input "Ausgangspos. und Anzahl der Zeichen eingeben"; S,N
  Mid$(V$,S,N)=T$
Print V$
Loop
```

=INSTR *(Sucht nach einer Zeichenfolge in einer anderen Zeichenkette)*

f=INSTR(d$,s$[,p])

Mit Hilfe von INSTR können Sie herausfinden, wo eine bestimmte Zeichenfolge in einer anderen Zeichenkette vorkommt. Dieser Befehl wird oft in Abenteuerspiele eingesetzt, um eine ganze Textzeile in die einzelnen Befehle zu zerlegen. Es gibt zwei Möglichkeiten, den INSTR-Befehl zu formulieren.

f=INSTR(d$,s$)

Dieser Befehl sucht nach dem ersten Vorkommen von s$ in d$. Wenn die Zeichenfolge gefunden ist, dann wird ihre Position sofort angezeigt, ansonsten wird das Ergebnis mit Null gleichgesetzt. Einige Beispiele dafür:

```
Print Instr("AMOS Basic","AMOS")
1

Print Instr("AMOS Basic","S")
4

Print Instr("AMOS Basic","AMIGA")
0

Do
   Input "Kette soll gesucht werden";D$
   Input "Kette soll gefunden werden";S$
   X=Instr(D$,S$)
   If X=0 Then Print S$;"Nicht gefunden"
   If X<>0 Then Print S$;"Gefunden an Position ";X
Loop
```

Normalerweise beginnt die Suche beim ersten Zeichen Ihrer Textfolge (d$). Mit der zweiten Version von INSTR können Sie jeweils einen bestimmten Abschnitt der Folge überprüfen. p stellt hier den Anfangspunkt Ihrer Suche dar. Alle Zeichen sind von links nach rechts, angefangen bei Null, numeriert. Deshalb geht p von 0 bis LEN(s$). Beispiele:

```
Print Instr("AMOS BASIC","S",0)
4

Print Instr("AMOS BASIC","S",5)
8
```

=UPPER$ *(Umsetzen einer Textfolge in Großbuchstaben)*

s$=UPPER$(n$)

Diese Funktion setzt die Zeichenfolge in n$ in Großbuchstaben um und setzt das Ergebnis dann in s$ ein. Hier ein Beispiel:

```
Print Upper$("AmOs BaSic")
AMOS BASIC
```

=LOWER$ *(Umsetzen einer Textfolge in Kleinbuchstaben)*

s$=LOWER$(n$)

LOWER$ schreibt alle Zeichen in n$ klein. Das ist bei Abenteuerspielen ganz besonders

nützlich, denn dann können Sie alles, was der Spieler eingibt, in ein Standard-Format bringen, das viel leichter zu interpretieren ist. Einige Beispiele:

```
Print Lower$("AMOS Basic")
amos basic
```

```
Input "Weitermachen (JA/NEIN)";ANSWER$
ANSWER$=Lower$(ANSWER$) : If ANSWER$="nein" Then Edit
Print "Weitermachen mit Ihrem Programm..."
```

=FLIP$ *(Kehrt eine Zeichenfolge um)*

f$=FLIP$(n$)

FLIP$ kehrt einfach die Zeichenfolge in n$ um. Das sieht dann zum Beispiel so aus:

```
Print Flip$("AMOS Basic")
cisaB SOMA
```

=SPACE$ *(Setzt Leerschritte in eine Zeichenkette ein)*

s$=SPACE$(n)

Generiert eine Folge von n Leerschritten und setzt sie in s$ ein. Dieser Befehl wird oft verwendet, um einen Textteil auszupolstern, bevor er auf dem Bildschirm angezeigt wird. Zum Beispiel:

```
Print "Zwanzig" ; Space$(20); "Leerschritte"
```

=STRING$ *(Erstellt ein Zeichenkette von a$)*

s$=STRING$(a$,n)

STRING$ erstellt eine Folge von n Ausführungen des ersten Zeichens in a$. Dafür ein Beispiel:

```
Print String$("Die Katze leckt die Tatze",10)
DDDDDDDDDD
```

Dabei ist STRING$(" ",N) identisch zu SPACE$(N).

=CHR$ *(Ausgabe als Ascii-Zeichen)*

s$=CHR$(n)

Erstellt eine Zeichenkette, die ein Zeichen mit dem Ascii-Code n enthält. Hier ein Beispiel:

```
For I=32 To 255 : Print Chr$(I); :Next I
```

Beachten Sie hier bitte, daß nur die Zeichen mit den Codes 32 bis 255 tatsächlich auf dem Bildschirm angezeigt werden können. Die anderen werden intern als Steuer-Codes eingesetzt. Bei den Textbefehlen wie zum Beispiel CUP$ finden Sie noch weitere Einzelheiten hierzu.

=ASC *(Gibt den Ascii-Code für ein Zeichen an)*

c=ASC(a$)

Der Befehl ASC liefert Ihnen den internen Ascii-Code für das erste Zeichen der Kette a$. Zum Beispiel:

```
Print Asc("B")
66
```

=LEN *(Ermittelt die Länge der Zeichenkette)*

l=LEN(a$)

Mit dem Befehl LEN können Sie die Länge der als a$ gespeicherten Zeichenfolge ermitteln. Zum Beispiel:

```
Print Len("12345678")
8
```

Bitte verwechseln Sie diesen Befehl nicht mit der LENGTH-Funktion, die man einsetzt, um die Länge einer AMOS-Speicherbank zu berechnen.

=VAL *(Wandelt eine Zeichenfolge in eine Zahl um)*

v=VAL(x$)
v#=VAL(x$)

VAL wandelt eine Reihe von in x$ gespeicherten Dezimalstellen in eine Zahl um. Sollte dieser Vorgang aus irgendeinem Grund nicht durchführbar sein, dann wird stattdessen der Wert Null ausgewiesen. Ein Beispiel:

```
X=Val("1234") : Print X
1234
```

=STR$ *(Wandelt eine Zahl in eine Zeichenkette um)*

s$=STR$(n)

STR$ wandelt eine Integervariable in eine Zeichenkette um. Das kann sehr nützlich sein, den einige Funktionen, wie CENTRE zum Beispiel, lassen die Eingabe von Zahlen als Parameter nicht zu. Beispiel:

```
Centre "Vorhandener Speicher ist"+Str$(Chip Free)+" Bytes"
```

Bitte nicht STR$ mit STRING$ verwechseln!

Array-Operationen

SORT *(Sortiert alle Elemente eines Arrays)*

SORT a(0)
SORT a#(0)
SORT a$(0)

Die SORT-Anweisung führt den Inhalt eines Arrays in aufsteigender Reihenfolge auf. Diese Gruppe kann entweder aus Zeichenketten, ganzen oder Fließkomma- Zahlen bestehen. Der a$(0) Parameter stellt den Anfangspunkt der Tabelle dar. Er muß immer auf den ersten Bestandteil des Arrays (Teil mit der Nummer Null) eingestellt sein. Beispiel:

```
Dim A(25)
P=0
Repeat
    Input "Eine Zahl eingeben (0 To Stop)";A(P)
    Inc P
Until A(P-1)=0 Or P>25
Sort A(0)
For I=0 To P-1
    Print A(I)
Next I
```

MATCH *(Sucht ein Array)*

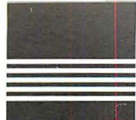
r=MATCH(t(0),s)
r=MATCH(t#(0),s#)
r=MATCH(t$(0),s$)

MATCH sucht ein sortiertes Array nach dem Wert s ab. Wenn die Suche erfolgreich war, dann wird die entsprechende Index-Nummer in r geladen. Ist die Suche aber vergeblich, dann wird das Ergebnis negativ sein. Der absolute Wert dieser Zahl nennt Ihnen den Bestandteil, der Ihrem ursprünglichen Such-Parameter am nächsten kam.

Auf diese Art können allerdings nur eindimensionale Arrays überprüft werden. Vergessen Sie nicht, das Array vorher zu sortieren. Hier ein Beispiel:

```
Read N
Dim D$(N)
For I=1 To N
    Read D$(I)
Next I
Sort D$(0)
Do
   Input A$
   If A$="Liste" Then For I=1 To N : Print D$(I) : Next I
   Exit if A$=""
   POS=Match(D$(0),A$)
   If POS>0 Then Print "Gefunden ",D$(POS);" Index"; POS
   If POS>0-N And POS=0
      Print "An nähsten ist : " ;
      Print D$ (Abs ( POS+1) )
   End If
Loop
Data 3, "MÜLLER", "SCHMIDT" , "ZOPT"
```

In Verbindung mit der INSTR-Funktion kann MATCH eine echt starke Parser-Routine bieten. Damit können Sie zum Beispiel die Anweisungen, die Sie in einem Adventure eingebaut haben, interpretieren.



6: Grafik

Mit AMOS-Basic können Sie die erstaunlichsten Grafik-Effekte hervorrufen. Sie finden hier den vollständigen Satz an Befehlen, den Sie zum Zeichnen von Rechtecken, Kreisen und Vielecken benötigen. Alle Anweisungen werden im Handumdrehen ausgeführt, aber etwas anderes erwarten Sie von Ihrem Amiga ja auch garnicht. Und sogar hier haben Sie mit AMOS-Basic noch einige Trümpfe in der Hand!

Die Grafikfunktionen von AMOS-Basic laufen in allen Grafikmodi des Amiga gleich gut, einschließlich des "Hold and Modify"- Modus (HAM). Deshalb können Sie direkt in AMOS-Basic atemberaubende HAM-Bilder kreieren.

Darüber hinaus müssen Sie sich nicht nur auf den sichtbaren Teil des Bildschirms beschränken. Wenn Sie eine besonders große Spielfläche geschaffen haben, dann können Sie über die Standard-Zeichenroutinen auf jeden der angezeigten Bereiche zugreifen. Auf diese Art ist das Erstellen von Scrolling-Hintergrund für Spielhallen- Spiele wie "Defender" ganz leicht.

Farben

Mit dem Amiga können Sie bis zu 64 Farben gleichzeitig auf dem Bildschirm darstellen. Diese Farben können mit dem Befehlen INK, COLOUR und PALETTE ausgewählt werden.

INK *(Wahl der Farbe für Zeichenschritte)*

INK col[,Papier][Umrandung]

col bestimmt die Farbe, die für alle folgenden Zeichenschritte eingesetzt wird. Die Farbe jedes Punktes auf dem Bildschirm wird aus 32 verschiedenen Farbregistern ausgewählt. Diese Register können mit einem Farbwert aus einer Palette von 4096 Farben ganz individuell belegt werden.

Der Amiga bietet Ihnen zwar nur 32 eigentliche Farbregister, aber mit AMOS können Sie Farbnummern von 0 bis 63 einsetzen. So können Sie die Farben des Half Bright bzw. HAM-Modus voll ausnützen. Im Kapitel "Bildschirme" geben wir Ihnen eine ausführliche Erklärung zu diesen Modi.

Die Farbe für das "Papier" bestimmt die Farbgebung der Füllmuster des Hintergrunds, die durch den Befehl SET PATTERN festgelegt werden. Die Farbgebung für die "Umrandung" bestimmt die Farbe, mit der Ihre Balken und Vielecke eingefaßt werden. Diese Option kann mit dem Befehl SET PAINT folgendermaßen aufgerufen werden:

```
Rem Zeichnet Formen von beliebiger Größe an beliebiger Position
Set pattern 0 : Set Paint 1
Repeat
   C=Rnd(16): Ink 16-C,0,C
   X=Rnd(320)-20 : Y=Rnd(200)-20 :S=Rnd(100)+10
```

```
    Bar X,Y To X+S,Y+S
  Until Mouse Key
```

Beachten Sie bitte, daß die Angabe der Parameter col, Papier und Umrandung auch weggelassen werden kann. Lassen Sie dazu einfach die leeren Kommas an der entsprechenden Stelle in der Anweisung stehen. Das kann zum Beispiel so aussehen:

```
  Ink,,5:Rem Bestimmt nur Farbe der Umrandung
```

COLOUR *(Weist einem Index eine Farbe zu)*

COLOUR Index,$RGB

Der Befehl COLOUR erlaubt Ihnen, jedem der 32 Farbregister des Amiga eine Farbe zuzuweisen. Index ist dabei die Nummer der Farbe, die Sie verändern möchten, und kann jede Zahl von 0 bis 31 sein. Wie Sie vielleicht schon wissen, kann jede Farbe erzeugt werden, indem man die Grundfarben Rot, Blau und Grün zu bestimmten Teilen mischt. Der Farbton, den Sie erhalten, hängt dabei völlig von der jeweiligen relativen Intensität der drei Bestandteile ab.

Die Hardware des Amiga ermöglicht Ihnen die Auswahl jeder Farbkomponente aus einer Palette von 16 verschiedenen Intensitätsstufen. So können Sie 16x16x16 (4096) verschiedene Farbtöne zusammenstellen. Es ist normalerweise üblich, diese Farben im Hexadezimalformat (Basis 16) anzugeben:

Hex.Stelle	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dezimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Der Ausdruck $RGB besteht aus drei Stellen der Zahlen und Buchstaben von 0 bis F. Jeder Bestandteil bestimmt die Stärke einer der Grundfarben Rot (R), Blau (B) oder Grün (G). Die Größe des Bestandteils ist direkt proportional zur Stärke der betreffenden Farbe. Also je höher der Wert, desto leuchtender die Farbe.

Hier sind ein paar Beispiele dafür:

Bestandteile	Hex Format	Farbergebnis
R=0 G=0 B=0	$000	Schwarz
R=F G=0 B=0	$F00	Hellrot
R=8 G=0 B=0	$800	Dunkelrot
R=F G=F B=0	$FF0	Gelb
R=F G=F B=0	$0F0	Grün
R=8 G=0 B=F	$80F	Violett
R=F G=F B=F	$FFF	Weiß
R=6 G=6 B=6	$666	Grau

Wenn Sie also Farbe Nummer 5 mit Gelb belegen möchten, dann geben Sie ein:

```
Colour 5,$FF0
```

Nach Ausführung dieser Anweisung wird jede auf dem Bildschirm angezeigte Grafik, in der die Farbe Nummer 5 verwendet wurde, in der neuen Farbe dargestellt. Beachten Sie bitte, daß die Modi HAM und Extra Half Bright diese Indizes etwas anders einsetzen. Nähere Einzelheiten dazu erfahren Sie in Kapitel 9.

=COLOUR *(Liest die Farbzuweisung)*

C=COLOUR(Index)

Die COLOUR-Funktion nimmt eine Index-Nummer von 0 bis 31 und zeigt den ihr zugewiesenen Farbwert an. Index ist dabei einfach die Nummer der Farbe, deren Schattierung Sie bestimmen möchten. Sie können mit dieser Funktion zum Beispiel folgendermaßen eine Liste der für Ihren Amiga gegenwärtig eingestellten Farben erstellen:

```
For C=0 to 15
  Print Hex$(Colour(C),3)
Next C
```

COLOUR BACK *(Bestimmen der Farbe des Hintergrundes)*

COLOUR BACK $RGB

Bestimmt die Farbe des Bildschirmhintergrundes. Die Farbe wird für die Bereiche des Bildschirms eingesetzt, die nicht verwendet werden, wie oben oder unten. Dazu folgendes Beispiel:

```
Colour Back Colour(0) : Rem Bestimmt eine Farbe für den gesamten Bildschirm
```

Die Hintergrundfarbe kann nicht ausgeblendet werden. Sie wird nur aktualisiert, wenn AMOS die Copperliste neu berechnet. Deshalb haben wir eine kleine Prozedur namens FAD_ALL eingebaut, durch die die Hintergrundfarben während einer FADE- Operation ganz glatt aktualisiert werden. Diese Prozedur finden Sie in der Datei Squash_Procs.AMOS auf der Extras-Diskette.

PALETTE *(Legt die aktuellen Farben auf dem Bildschirm fest)*

PALETTE Liste von Farben

Die Anweisung PALETTE ist eigentlich nur eine verstärkte Version des Befehls COLOUR. Anstatt die Farbwerte einen nach dem anderen zu laden, können Sie über PALETTE mit einer einzigen Anweisung eine ganze Palette neuer Farben laden.

Sie müssen dabei aber nicht die gesamte Palette definieren. Jede beliebige Kombination von Farben kann einzeln geladen werden, zum Beispiel:

```
Palette $100,$200,$300 : Rem Bestimmt nur drei Farben
```

Sie können auch bestimmte Farben in der Mitte Ihrer Liste folgendermaßen verändern:

```
Palette $200,,$400 : Rem Verändert die Farben 0 und 2
```

Sie dürfen dabei nicht vergessen, daß nur die Farben in der Palette, die durch diese Anweisung speziell angesprochen werden, auch verändert werden. Für alle anderen Farben werden die ursprünglich festgelegten Werte beibehalten. Hier sind einige Beispiele:

```
Palette 0,$F00,$0F0
Palette 0,$770
Palette 0,,$66
Palette 0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$A,$B,$C,$D,$E,$F
```

Am Anfang Ihres Programms wird die Farbpalette automatisch mit einer Reihe von Default-Werten geladen. Diese Parameter können mit einer einfachen Option aus dem AMOS Konfigurations-Programm verändert werden. Mit diesem Befehl können Sie auch die Farben der Half Bright und HAM Modi einstellen. Diese Modi erweitern die bestehende Farbpalette um eine Vielzahl weiterer Farbtöne. In Kapitel 10 finden Sie dazu weitere Informationen.

Befehle zum Ziehen von Linien

GR LOCATE *(Positioniert den Grafik-Cursor)*

GR LOCATE x,y

Mit diesem Befehl wird der Grafik-Cursor auf dem Bildschirm an den Koordinaten x,y positioniert. Der Grafik-Cursor dient als Default-Anfangspunkt für die meisten Zeichenschritte. Wenn Sie also die Koordinaten bei Befehlen wie PLOT oder CIRCLE nicht angeben, dann werden die Objekte an der aktuellen Cursor-Position gezeichnet. Zum Beispiel:

```
Gr Locate 10,10 : Plot ,
Gr Locate 100,100 : Circle ,,100
```

=XGR *(Gibt X-Koordinate des Grafik-Cursors an)*
=YGR *(Gibt Y-Koordinate des Grafik-Cursors an)*

x=XGR
y=YGR

Mit diesen Funktionen können Sie die gegenwärtigen Koordinaten des Grafik- Cursors ermitteln. Zum Beispiel:

```
Circle 10,100,100
Print Xgr,Ygr
```

PLOT *(Plottet einen einzigen Punkt)*

PLOT x,y [,c]

Der PLOT-Befehl ist die einfachste Zeichenfunktion in AMOS-Basic. Sie plottet einen Punkt an den Koordinaten x,y in der Farbe c. Diese neue Farbe wird auch für alle folgenden Zeichenschritte verwendet. Wenn bei dieser Anweisung die Farbe c nicht angegeben wird, denn wird der Punkt in der gegenwärtig gewählten Farbe geplottet. Zum Beispiel:

```
Curs Off : Flash Off : Randomize Timer
Do
 Plot Rnd(319),Rnd(199),Rnd(15)
Loop
```

Sie können diese Anweisung natürlich auch ohne die Angabe der X- und Y- Koordinaten eingeben. Dann wird der Punkt an der derzeitigen Cursor-Position geplottet.

```
Plot 100,100,4
Plot ,150
Cls : Plot ,
```

POINT *(Gibt die Farbe eines Punktes an)*

C=POINT(x,y)

Mit dem Befehl POINT können Sie den Farbindex eines Punktes mit den Koordinaten x,y ermitteln. Hier ist ein Beispiel:

```
Plot 100,100
Print "Die Farbe bei 100,100 ist ";Point(100,100)
```

DRAW *(Zieht eine Linie)*

DRAW ist ein einfacher, grundlegender Befehl, dessen Funktion es ist, auf dem Bildschirm des Amiga eine einfache, gerade Linie zu ziehen.

DRAW x1,y1 TO x2,y2

Zieht eine Linie zwischen den Koordinaten *x1,y1* und *x2,y2*.

DRAW TO x3,y3

Zieht eine Linie von der aktuellen Cursor-Position zu den Koordinaten *x3,y3*. Zum Beispiel:

```
Colour $707 : Ink 4
Draw 0,50 To 200,50
Draw To 100,100
Draw To 0,50
```

Siehe dazu auch die Befehle POLYLINE, INK.

BOX *(Zeichnet ein leeres Rechteck auf den Bildschirm)*

BOX x1,y1 TO x2,y2

Mit dem Befehl BOX können Sie auf dem Bildschirm Ihres Amiga ein leeres Rechteck erzeugen. Dabei stellen *x1,y1* die Koordinaten der linken oberen Ecke und *x2,y2* die Koordinaten des diagonal gegenüberliegenden Punktes dar. Zum Beispiel:

```
Curs Off : Flash Off : Randomize Timer
Do
 Ink Rnd(15)
 X1=Rnd(320) : Y1=Rnd(200) : Box X1,Y1 To X1+Rnd(50),Y1+Rnd(50)
Loop
```

Siehe auch die Befehle SET LINE, INK und BAR.

POLYLINE *(Zieht mehrere Linien)*

POLYLINE ist dem Befehl DRAW sehr ähnlich, hier werden jedoch mehrere Linien gleichzeitig gezogen. So können Sie mit einer einzigen Anweisung komplexe leere Vielecke erstellen.

POLYLINE x1,y1 TO x2,y2, TO x3,y3 ...
POLYLINE TO x1,y1, TO x2,y2 ...

Dabei sind x1,y1 die Koordinaten von Punkt 1; *x2,y2* von Punkt 2 und *x3,y3* von Punkt 3.

Mit dem Befehl POLYLINE wird eine Linie zwischen jedem Koordinatenpaar auf Ihrer Liste gezogen. Die erste Linie verläuft also von Punkt 1 zu Punkt 2, die zweite von Punkt 2 zu Punkt 3 usw.

Das entspricht den folgenden Anweisungen:

Draw x1,y1 TO x2,y2
Draw To x3,y3
Draw To x4,y4

Hier ist ein einfaches Beispiel, mit dem Sie ein Dreieck auf dem Amiga Bildschirm entstehen lassen können:

```
Polyline 0,20 To 200,20 To 100,100 To 0,20
```

Siehe hierzu auch die Befehle SET LINE, INK und POLYGON.

CIRCLE *(Zieht einen leeren Kreis)*

CIRCLE x,y,r

Der Befehl CIRCLE zieht einen leeren Kreis mit dem Radius *r* und dem Mittelpunkt *x,y*. Zum Beispiel:

```
Curs Off : Flash Off : Randomize Timer
Do
 Ink Rnd(15)
 X=Rnd(200) : Y=Rnd(100) : R=Rnd(90) : Circle X,Y,R
Loop
```

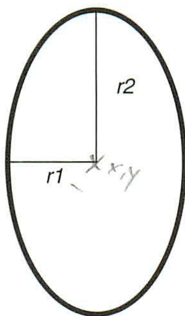
Wie üblich können die Koordinaten auch bei diesem Befehl weggelassen werden, dann wird der Kreis an der aktuellen Cursor-Position gezogen.

ELLIPSE *(Zeichnet eine leere Ellipse)*

ELLIPSE x,y,r1,r2

Mit dem Befehl ELLIPSE können Sie eine leere Ellipse an den Koordinaten *x,y* zeichnen. *r1* ist dabei der horizontale Radius und entspricht genau der halben Breite der Ellipse. *r2* ist der vertikale Radius und bestimmt die Höhe der Ellipse. Die Gesamthöhe der Ellipse beträgt *r*2*.

Die Radien einer Ellipse




```
Curs Off : Flash Off : Randomize Timer
Do
  Ink Rnd(15) : X=Rnd(200) : Y=Rnd(100) : R1=rnd(90) : R2=rnd(90)
  Ellipse X,Y,R1,R2
Loop
```

Linienarten

AMOS-Basic bietet Ihnen eine Vielzahl von Stilarten für die Gestaltung Ihrer Linien.

SET LINE *(Bestimmt die Stilarten)*

SET LINE Maske

Mit dem Befehl SET LINE bestimmen Sie, in welchem Stil alle folgenden Linien gestaltet werden, die Sie mit den Befehlen DRAW, BOX oder POLYLINE ziehen. Maske ist dabei eine binäre 16-Bit-Zahl, die das genaue Erscheinungsbild der Linie beschreibt. Die Punkte der Linie, die in der aktuellen Zeichenfarbe (INK colour) dargestellt werden

sollen, werden durch eine 1 ausgedrückt, und die Punkte, die in der Farbe des Hintergrunds erscheinen sollen, mit einer Null. So wird eine normale, durchgezogene Linie mit der Binärzahl %1111111111111 beschrieben, und wird dann so dargestellt: ______________. Entsprechend wird diese unterbrochene Linie: _ _ _ _ _ _ _ durch die Maske %111100001111000011110000 ausgedrückt.

Durch Setzen der Linienmaske auf Werte zwischen 0 und 65535 kann man eine Vielzahl verschiedener Gestaltungsmöglichkeiten wählen. Zum Beispiel:

```
Set Line $F0F0
Box 50,100 To 150,150
```

$ F0F0 = gleichmäßig
% 111000 = un- "

Diese Stilart wird nur zum Ziehen von Linien mit dem DRAW Befehl eingesetzt, und hat auf Formen, die mit dem Befehlen CIRCLE oder ELLIPSE erstellt werden, keinen Einfluß.

Füllmuster

PAINT *(Füllen von Konturen)*

PAINT x,y Modus

0 = Stop bei Umrandung
1 = bei Abweichung von INKcolour

Mit dem Befehl PAINT können Sie jeden beliebigen Bereich auf dem Bildschirm mit einem Farbblock füllen. Außerdem können Sie mit dem Befehl SET PATTERN die Füllmuster für Ihre Formen bestimmen. *x,y* sind die Koordinaten eines Punktes innerhalb des zu füllenden Bereichs. Modus kann entweder auf 0 oder 1 gesetzt werden. Ein Wert von 0 beendet das Füllen beim ersten Pixel mit der Farbe der Umrandung. Der Modus 1 stoppt das Füllen beim ersten Pixel jeder Farbe, die von der gegenwärtig für die Zeichenschritte eingegebenen Farbe (INK colour) abweicht.

Mit PAINT können Sie jede beliebige Fläche auffüllen, vorausgesetzt, sie ist vollständig umrandet. Wenn diese Umrandung jedoch nicht völlig geschlossen ist, dann läuft die Farbe aus und füllt den Rest des Bildschirms. **BEISPIEL 6.1** im MANUAL-Unterverzeichnis demonstriert dies.

BAR *(Zeichnet ein ausgefülltes Rechteck)*

BAR x1,y1 TO x2,y2

Dieser Befehl zeichnet einen ausgefüllten Balken von den Koordinaten *x1,y1* - der Ecke links oben - zu den Koordinaten *x2,y2*, der diagonal gegenüberliegenden Ecke. Zum Beispiel:

```
Curs Off : Flash Off : Randomize Timer
Do
   X1=rnd(200) : Y1=rnd(100) : W=rnd(100) : H=rnd(80)
   Ink Rnd(15) : Bar X1,Y1 To X1+W,Y1+H
Loop
```

Siehe auch die Befehle BOX, SET PAINT und INK.

POLYGON *(Zeichnet ein ausgefülltes Vieleck)*

POLYGON x1,y1 TO x2,y2 TO x3,y3 ...
POLYGON TO x1,y1 TO x2,y2 ...

Mit dem Befehl POLYGON können Sie ein ausgefülltes Vieleck in der gegenwärtig für das Ziehen von Linien gewählten Farbe erstellen. Im Grunde handelt es sich hier nur um eine Abwandlung des Befehls POLYLINE. Sie können auch hier wie gehabt die Farbe mit dem INK-Befehl und das Füllmuster mit dem PAINT-Befehl bestimmen.

Die Koordinaten *(x1,y1)*, *(x2,y2)* und *(x3,y3)* markieren die Anfangs- und Endpunkte der Linien, aus denen das Vieleck besteht. Die Anzahl der Koordinatenpaare ist hierbei eigentlich nicht beschränkt und nur durch die maximale Zeilenlänge in AMOS-Basic - nämlich 255 Zeichen - begrenzt. Sie sehen also, daß Sie mit diesem Befehl ganz schön komplexe Gebilde schaffen können.

Es gibt auch vor diesen Befehl eine zweite Version, durch die das Vieleck an der aktuellen Cursor-Position begonnen wird. Sie hat folgendes Format:

POLYGON TO x1,y1 TO x2,y2

Abgesehen von den Koordinaten für den Anfangspunkt stimmt diese Anweisung mit dem normalen POLYGON-Befehl überein.

```
Do
   Ink Rnd(15)
   X1=Rnd(200) : Y1=Rnd(100) : H=Rnd(100) : W=Rnd(90)
   Polygon X1,Y1 To X1+W,Y1 To X1+W/2,Y1+H To X1,Y1
Loop
```

Das obengenannte Programm füllt den Bildschirm mit hübschen, bunten Dreiecken. Schlagen Sie dazu auch die Befehle POLYLINE, INK und SET PAINT nach.

Füllmuster

AMOS-Basic gibt Ihnen aber nicht nur die Möglichkeit, Ihre geometrischen Formen mit Farbe zu füllen, sondern bietet Ihnen unzählige Füllmuster zur Auswahl. Sie können außerdem auch Ihre eigenen Muster direkt aus der Sprite-Bank laden.

SET PATTERN *(Auswahl eines Füllmusters)*

SET PATTERN Muster

Mit diesem Befehl können Sie ein Füllmuster für Ihre nächsten Zeichenschritte auswählen. Es gibt drei Möglichkeiten:

pattern=0

Das ist der Default, so werden Ihre Formen mit einem durchgängigen Block in der gegenwärtig gewählten Farbe ausgefüllt.

pattern>0

Wenn für das Muster ein Wert, der größer als Null ist, eingegeben wird, dann wählt AMOS-Basic eines der 34 eingebauten Füllmuster. Diese Muster befinden sich in der Datei MOUSE.ABK auf Ihrer Startdiskette und können mit dem AMOS Sprite Definer bearbeitet werden. Beachten Sie dabei bitte, daß die ersten drei Bilder dieser Datei vom Maus-Cursor benötigt werden (siehe Befehl CHANGE MOUSE). Die Füllmuster sind in den Bildern ab Nummer vier gespeichert.

pattern<0

Dies ist die stärkste aller Optionen, denn jetzt bezieht sich pattern auf ein Sprite- Bild in Bank eins. Die Nummer des Bildes wird mit folgender Formel berechnet:

SPRITE IMAGE = PATTERN *-1

Das gewählte Bild wird automatisch vor dem Einsatz nach folgenden Regeln bearbeitet:

- Die Breite des Bilds wird auf 16 Pixel beschränkt.
- Die Höhe wird auf die nächstliegende Potenz von 2 gerundet, z.B. 1, 2, 4, 8, 16, 32, 64.

Je nach Art Ihres Bildes wird das Muster auf eine der beiden folgenden Weisen gezeichnet. Zweifarbige Bilder werden in Monochrome dargestellt. Die tatsächlichen Farben Ihres Bildes werden völlig vernachlässigt, und das Muster wird in den gegenwärtigen Farben für Zeichnung und Hintergrund erstellt.
Sie können aber auch bunte Füllmuster hervorrufen. In diesem Fall werden die Farben für den Vordergrund Ihres Bildes mit der aktuellen Zeichenfarbe durch ein logisches AND verbunden. Die Farbe des Bildhintergrunds wird auf ähnliche Weise durch OR mit dem Sprite-Hintergrund (Farbe Null) abgeglichen. Wenn Sie stattdessen Ihre ursprünglichen Sprite-Farben einsetzen möchten, dann müssen Sie die Farben für Bild und Hintergrund folgendermaßen angeben:

```
Ink 31,0
```

Vergessen Sie aber nicht, mit dem Befehl GET SPRITE PALETTE Ihre Sprite-Palette aus der Sprite-Bank zu laden, bevor Sie diese Anweisungen eingeben, sonst sieht es auf Ihrem Bildschirm nachher wahrscheinlich ziemlich wüst aus. Beispiele für die Anwendung dieser Befehle finden Sie in **BEISPIEL 6.2** im MANUAL- Unterverzeichnis. Siehe auch Befehle CIRCLE, ELLIPSE, BAR und POLYGON.

SET PAINT *(Einstellen/Verändern der Umrandung)*

SET PAINT n

Der Befehl SET PAINT schaltet die verschiedenen Arten der Umrandung Ihres Vielecks oder Balkens ein und aus. Dieser Modus ist als Default auf AUS gestellt. Wenn n=1 ist, dann wird der Modus für die Umrandung eingeschaltet, und es wird um Ihre geometrische Form eine Linie in der Farbe gezogen, die gerade für die Umrandung gewählt ist. Zum Beispiel:

```
Ink ,,5 : Set Paint 1 : Bar 100,100 to 200,150
```

Sie können das Umranden wieder ausschalten, indem Sie für SET PAINT einen Wert von Null eingeben.

Schriftarten

GR WRITING *(Wechselt die Schriftart)*

GR WRITING Bitmuster

Wenn Sie auf dem Amiga-Bildschirm Grafiken erstellen, dann gehen Sie natürlich davon aus, daß alles, was unter der Grafik liegt, überschrieben wird. Mit dem Befehl GR WRITING können Sie zwischen vier verschiedenen Stilarten wählen und so Dutzende von erstaunlichen Effekten schaffen. Bitmuster enthält eine Folge von binären Bits, die angeben, in welchem Grafik-Modus Sie arbeiten möchten. Hier sind nun die verschiedenen Möglichkeiten zusammen mit einer kurzen Erklärung ihrer Wirkung aufgeführt.

JAM1 Modus *Bit 0=0*
JAM1 zeichnet nur die Teile Ihrer Grafik, die in der gegenwärtig gewählten Farbe erstellt wurden. Alle Teile, die in der Farbe des Hintergrunds gezeichnet wurden, werden überhaupt nicht dargestellt. Das ist vor allem in Verbindung mit dem TEXT- Befehl sehr nützlich, weil Sie so Ihren Text direkt über einen bereits vorhandenen Hintergrund einsetzen können. Zum Beispiel:

```
Ink 2,5 : Text 140,80,"Normaler Text" : Gr Writing 0 : Text 140,71,"JAM1"
```

JAM2 Modus *Bit 0=1*
Dies ist der Default. Jede vorhandene Grafik auf dem Bildschirm wird durch Ihr neues Bild völlig überschrieben.

XOR Modus *Bit 1=1*
XOR verbindet Ihre neue Grafiken mit den bereits auf dem Bildschirm vorhandenen mittels eines logischen Schritts, den man als "eXklusive OR" bezeichnet. Das Endergebnis besteht in der Veränderung der Farbe der Bereiche einer Zeichnung, die sich mit einem bereits vorhandenen Bild überschneiden.

Eine interessante Nebenwirkung des XOR-Modus besteht darin, daß Sie jedes beliebige Objekt vom Bildschirm löschen können, indem Sie den XOR-Modus einschalten und dasselbe Objekt an der gleichen Position noch einmal zeichnen. **BEISPIEL 6.3** gibt Ihnen eine einfache kurze Demonstration dieser Technik und produziert einen echt irren Gummiband-Effekt.

INVERSEVID *Bit 2=1*
Hier wird das Bild vor dem Zeichnen umgekehrt. Alle Teile der Zeichnung, die in der Schriftfarbe waren, werden in der gegenwärtig gewählten Hintergrundfarbe dargestellt und umgekehrt. INVERSEVID wird meist dazu eingesetzt, Text hervorzuheben.

Da diese Modi unter Verwendung eines Bitmusters eingegeben werden, können mehrere miteinander verbunden werden.

```
Gr Writing 4+1 : Rem Einstellen von JAM2 und INVERSEVID
Gr Writing 4+2+1 : Rem Wählt JAM2, INVERSEVID und XOR
Ink 2,5 : Text 140,80,"Normaler Text"
Gr Writing 5 : Text 140,71,"Inversevid+Jam2"
```

Achtung: Dieser Befehl wirkt nur auf Zeichenschritte mit den Anweisungen CIRCLE, BOX und Textgrafik (TEXT). Der Zeichenmodus, der beim Einsatz der normalen Textbefehlen wie PRINT und CENTRE verwendet wird, wird mit einem eigenen WRITING-Befehl aufgerufen. Siehe dazu auch Befehle AUTOBACK und WRITING.

CLIP *(Begrenzt alle Grafiken auf einen Teil des Bildschirms)*

CLIP [x1,y1 TO x2,y2]

Der Befehl CLIP beschränkt alle Zeichenschritte auf einen rechteckigen Bereich des Bildschirms, der durch die Koordinaten *x1,y1* bis *x2,y2* markiert ist. *x1,y1* stellen dabei die Koordinaten der linken oberen Ecke des Rechtecks, und *x2,y2* die Koordinaten der rechten unteren Ecke dar. Beachten Sie bitte, daß Sie ohne weiteres Koordinaten angeben können, die außerhalb des normalen Bildschirmbereiches liegen. Alle Clip-Vorgänge können wie normal durchgeführt werden, auch wenn nur ein Teil des Rechtecks tatsächlich sichtbar ist.

Ein ausführliches Beispiel für diesen Befehl finden Sie in **BEISPIEL 6.4** im MANUAL-Unterverzeichnis. Wie Sie sehen, werden nur die Teile des Kreises, die im

MANUAL-Unterverzeichnis. Wie Sie sehen, werden nur die Teile des Kreises, die im Bereich des Rechtecks liegen, auf dem Bildschirm angezeigt. Sie können den ganzen Schirm wieder als Arbeitsbereich definieren, indem Sie den CLIP-Befehl nochmals eingeben, jedoch ohne Angabe von Koordinaten.

Techniken für Fortgeschrittene

SET TEMPRAS *(Erstellt ein temporäres Raster)*

SET TEMPRAS [Adresse, Größe]

Mit dieser Anweisung können erfahrene Amiga-Programmierer eine Feineinstellung des Speicherplatzes, der von den jeweiligen Grafikoperationen eingenommen wird, durchführen. **Achtung!** Wenn Sie diesen Befehl falsch einsetzen, kann Ihr Amiga total abstürzen!

Wenn ein AMOS-Programm einen Füllbefehl ausführt, dann wird ein besonderer Speicherbereich reserviert, um das Füllmuster aufzunehmen. Dieser Speicher wird dem System automatisch wieder zugeführt, nachdem die Anweisung durchgeführt wurde.

Die Größe des Pfufferspeichers entspricht einer einzigen Bitplane im gegenwärtig eingestellten Bildschirm-Modus. So nimmt der Default-Bildschirm zum Beispiel im Ganzen 8KB ein.

Größe und Bereich des Grafikpuffers können jederzeit mit dem SET TEMPRAS Befehl verändert werden. Die Größe ist dabei die Anzahl der Bytes, die Sie für Ihren Pufferbereich reservieren möchten. Sie kann einen beliebigen Wert zwischen 256 und 65536 betragen.

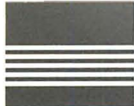
Den Speicherplatz, den ein bestimmtes Objekt benötigt, kann man folgendermaßen berechnen:

- Ziehen Sie um das betreffende Objekt mit einem rechteckigen Rahmen.
- Danach wird der erforderliche Platz folgendermaßen ermittelt:Größe=Breite/8*Höhe

Wenn Sie vorhaben, den Befehl PAINT einzusetzen, dann achten Sie darauf, daß Ihre geometrische Form vor dem Füllen völlig geschlossen ist, sonst läuft die Farbe aus, es wird mehr Speicher benötigt, und das System stürzt womöglich noch ab!

Für den Puffer kann entweder ein Label oder eine Bank angegeben werden. Der Speicher, den Sie für diesen Puffer reservieren, sollte immer Chip-RAM sein. Da der Pufferbereich jetzt am Beginn Ihres Programmes ein für allemal festgelegt ist, besteht keine Notwendigkeit mehr, den Pufferspeicher immer wieder zu reservieren und aufzurufen. Damit kann die Ausführung Ihrer Programme um bis zu 5% beschleunigt werden.

Sie können den Pufferbereich wieder auf seine ursprüngliche Größe bringen, indem Sie den SET TEMPRAS Befehl ohne Angabe der Parameter noch einmal eingeben. **BEISPIEL 6.5** im MANUAL-Unterverzeichnis zeigt Ihnen, wie dieser Befehl eingesetzt werden kann.



7: Kontrollstrukturen

Die meisten modernen Programmiersprachen enthalten eine Reihe von Anweisungen, die es Ihnen ermöglichen, in Ihren Programmen Entscheidungen zu treffen und Schleifen zu fahren. Diese Anweisungen nennt man Kontrollstrukturen. AMOS bietet Ihnen die gesamte Palette der Basic-Kontrollstrukturen. Hier finden Sie alle Ihre Lieblingsbefehle wie GOTO und FOR ... NEXT wieder und dazu noch eine Reihe interessanter neuer Varianten wie zum Beispiel den Befehl ON ... EVERY.

GOTO *(Springt zu einer neuen Zeilennummer)*

In den schlechten alten Zeiten des Computerns wurde GOTO wohl von allen Basic-Anweisungen am häufigsten eingesetzt. Heutzutage bezeichnet man sie oft als alten Hut und ersetzt sie durch strukturierte Befehle wie DO ... LOOP und IF ... ELSE ... ENDIF. Diese Anweisungen sind normalerweise übersichtlicher, und wir würden Ihnen auch empfehlen, sie wenn möglich anstelle von GOTO zu verwenden. GOTO dient dazu, die Steuerung des Programms von einem Standort zu einem anderen zu verlagern. AMOS-Basic läßt drei Varianten des GOTO Befehls zu.

GOTO Sprungmarke

Die Sprungmarke ist eine frei wählbare Markierung seitlich an einer Zeile. Die Namen der Sprungmarke werden unter Verwendung des Doppelpunktes ":" folgendermaßen definiert:

Sprungmarke:

Der Name der Sprungmarke kann aus einer beliebigen Folge alphanumerischer Zeichen bestehen, einschließlich des Bindestrichs "-". Es gelten dabei dieselben Regeln wie für die Namen der Variablen und Prozeduren.

GOTO Zeilennummer

Jeder Zeile in AMOS-Basic kann wahlweise eine Nummer vorangestellt werden. Diese Zeilennummern dienen in erster Linie der Kompatibilität zu anderen Basic- Versionen (wie zum Beispiel STOS und Atari ST). Es ist aber besser, sich stattdessen auf Sprungmarken zu verlassen, da sie viel leichter zu lesen und zu behalten sind.

GOTO Variable

Als Variable kann hier jeder erlaubte AMOS Basic-Ausdruck gelten. Dieser Ausdruck kann entweder ein normaler ganzzahliger Wert oder eine Zeichenfolge sein. Ganze Zahlen geben dabei die Zeilennummer für Ihren GOTO-Befehl, Zeichenketten (Strings) die entsprechende Sprungmarke an.

In der Fachsprache bezeichnet man diese Konstruktion als berechnetes Goto. Die meisten ernsthaften Programmierer rümpfen darüber die Nase, aber manchmal kann dieser Befehl ganz ungemein nützlich sein. Hier sind einige Beispiele:

```
RAUM=3
BEGINN:
Goto "RAUM"+Str$(RAUM)-""
End
RAUM3:
Print "Raum drei!"
Goto BEGINN
```

Siehe dazu auch den Befehl ON GOTO.

GOSUB *(Springt zu einer Subroutine)*

GOSUB ist eine weitere veraltete Anweisung, die Ihnen die angenehme Möglichkeit bietet, ein Programm in kleinere, übersichtlichere Teile aufzuspalten, die man auch Subroutinen nennt. Mittlerweile ist der Befehl GOSUB fast völlig vom AMOS Basic-System der Prozeduren ersetzt worden. Aber GOSUB stellt immer noch ein nützliche Übergangslösung dar, wenn Sie Programme aus einer anderen Basic- Version wie STOS konvertieren.

Wie bei GOTO gibt es auch für GOSUB drei Varianten:

GOSUB n	Springt zur Subroutine in Zeile n.
GOSUB Name	Springt zu einer AMOS-Sprungmarke.
GOSUB Ausdruck	Springt zu der Zeilennummer oder der Sprungmarke, das durch den Ausdruck bezeichnet wird.

Beispiel:

```
For I=1 To 10
   Gosub TEST
Next I
Direct
TEST:
Print "Das ist ein Beispiel für GOSUB" : Print "I ist gleich ";I
Return : Rem Ausstieg aus Subroutine TEST und zurück zum Hauptprogramm
```

Es ist dabei ganz ratsam, Ihre Subroutinen an das Ende Ihres Hauptprogramms zu packen, weil sie so leichter aus Ihrem Programmverzeichnis herausgesucht werden können. Sie sollten am Ende Ihres Hauptprogrammes auch eine Anweisung wie Edit oder Direct anhängen, da AMOS sonst versucht, Ihre GOSUB-Befehle noch auszuführen, wenn das Programm schon zu Ende ist, und so eine Fehlermeldung hervorruft.

RETURN *(Zurück ins Hauptprogramm)*

RETURN

Der Befehl RETURN bewirkt die Rückkehr zum Hautprogramm aus einer Subroutine, die durch GOSUB aufgerufen wurde. Mit diesem Befehl springt man sofort zur nächsten Basic-Anweisung nach dem ursprünglichen GOSUB-Befehl zurück. Beachten Sie bitte, daß eine GOSUB-Anweisung mehrere RETURN-Befehle umfassen kann. Je nach Situation können Sie von jedem beliebigen Punkt aus Ihrer Routine aussteigen.

POP *(Entfernt die RETURN-Anweisung nach einem GOSUB)*

POP

Normalerweise ist das Aussteigen aus einer GOSUB-Anweisung mit dem normalen GOTO nicht erlaubt. Das kann manchmal ziemlich hinderlich sein, vor allem wenn sich ein Fehler eingeschlichen hat und Sie jetzt nicht mehr an dieselbe Stelle in Ihr Programm zurückgehen können, an der Sie vorher waren.

Mit dem POP-Befehl können Sie jetzt die Zieladresse, die von Ihrem GOSUB generiert wurde, löschen und frei wählen, wie Sie in Ihre Subroutine zurückgehen wollen. Sie müssen jetzt nicht erst den abschließen RETURN-Befehl ausführen. Hier ein Beispiel:

```
Do
   Gosub TEST
Loop
TSCHUESS:
Print "Ausgestiegen"
Direct: Rem Subroutinen vom Hautprogramm getrennt halten
TEST:
Print "Hallo!"
If Mouse Key Then Pop : Goto TSCHUESS
Return
```

Siehe hierzu auch den Befehl ON GOSUB.

IF...THEN...[ELSE] *(Wählen zwischen verschiedenen Aktionen)*

Mit der Anweisung IF..THEN können Sie innerhalb eines Basic-Programms einfache Entscheidungen treffen. Das Format lautet:

IF Bedingung THEN Anweisung 1 [ELSE Anweisung 2]

Die Bedingung kann jede beliebige Reihe von Tests, einschließlich AND und OR, darstellen. Bei Anweisung 1 und Anweisung 2 muß es sich um eine Reihe von AMOS Basic-Anweisungen handeln. Wenn Sie zu einer Zeile oder einer Sprungmarke springen möchten, dann müssen Sie einen eigenen GOTO-Befehl folgendermaßen miteinbeziehen:

If test Then GOTO Label : Rem Das ist ok

Wenn Sie das vergessen und versuchen, die GOTO-Anweisung wie im normalen Basic zu umgehen, dann behandelt AMOS Ihre Sprungmarke wie einen Prozedur- Namen, und Sie erhalten die Fehlermeldung Procedure nicht definiert.

If test Then Label : Rem Das ruft eine PROZEDUR auf

Der Bereich für diese IF...THEN Anweisung ist auf eine einzige Zeile in Ihrem Basic-Programm begrenzt. Deshalb wird sie jetzt von dem viel stärkeren IF...ELSE...ENDIF Befehl abgelöst.

IF...[ELSE]...ENDIF *(Strukturierter Test)*

Obwohl die ursprüngliche Form von IF..THEN zweifelsohne nützlich ist, erscheint sie im Vergleich zu den Möglichkeiten, die die modernen Basic-Versionen wie zum Beispiel AMOS bieten, doch ziemlich altbacken. Jetzt können Sie stattdessen eine ganze Reihe von Anweisungen ausführen, je nachdem, wie das Ergebnis eines einzigen Tests ausfällt.

```
IF tests=TRUE
        Auflistung der Anweisungen 1
        :           :           :
ELSE
        Auflistung der Anweisungen 2
        :           :           :
ENDIF
```

Sie können hier jeweils jede beliebige Gruppe von AMOS Basic-Anweisungen auflisten, einschließlich weiterer IF...ENDIF Befehle. Es ist jedoch **nicht erlaubt,** ein normales IF..THEN in einem strukturierten Text einzusetzen. Diese Befehle sollten durch die entsprechende IF...ENDIF Anweisung folgendermaßen ersetzt werden:

If test Then Goto Label Else Label2

Das sieht dann so aus:

If test : Goto Label : Else goto Label2 : Endif

oder:

```
If test
  Goto Label
Endif
```

Hier ist ein Beispiel, an dem Sie den IF..ENDIF Befehl in Aktion sehen können:

```
Input "Eingabe Werte für a,b, und c";A,B,C
If A=B
   Print "Gleich"
Else
   Print "Verschieden";
   If A<>B und A<>C
        Print ",und C ist auch nicht gleich!"
   End If
End If
```

Jede IF-Anweisung in Ihrem Programm **muß** mit einem ENDIF-Befehl gepaart sein, da dies AMOS-Basic genau anzeigt, welche Gruppe von Anweisungen in Ihrem Test ausgeführt werden sollen. Beachten Sie hier bitte, daß "THEN" in dieser Fassung des Befehls überhaupt nicht eingesetzt wird. Das ist vielleicht etwas gewöhnungsbedürftig, wenn Sie schon mit den anderen Basic-Versionen für den Amiga vertraut sind.

Siehe auch die Befehle AND, OR, NOT, TRUE, FALSE.

FOR...NEXT *(Wiederholt einen Code-Abschnitt)*

Dies ist die klassische Methode zur Wiederholung von Abschnitten Ihres Basic-Programms. Das Format der Anweisung lautet folgendermaßen:

```
FOR Index=Anfang TO Ende [STEP inc]
 Liste der Anweisungen
:      :    :
NEXT[Index]
```

Mit dem Befehl FOR...NEXT können Sie angeben, wie oft Ihre Liste wiederholt werden soll. Index enthält einen Zähler, der nach jeder einzelnen Schleife (Loop) zunimmt. Am Anfang der Schleife wird das Ergebnis des Ausdrucks in Anfang in diesen Zähler geladen. Danach werden die Anweisungen, die zwischen FOR und NEXT eingegeben wurden, ausgeführt, bis das Programm bei NEXT ankommt.

inc stellt einen Wert dar, der nach jeder Schleife durch die NEXT-Anweisung zum Zähler hinzugefügt wird. Wenn Sie hier nichts eingeben, dann wird der Wert automatisch auf "1" gesetzt. Nachdem dieser Zähler durch den Befehl NEXT auf den neuesten Stand

gebracht wurde, wird überprüft, ob der aktuelle Wert größer als der vorhergehende ist. Ist das der Fall, so wird die Schleife sofort beendet und die direkt auf NEXT folgende Anweisung ausgeführt. Ansonsten wird die Schleife noch einmal von vorn begonnen.

Wurde für inc ein negativer Wert eingesetzt, dann wird die Schleife angehalten, sobald der Wert im Zähler geringer als der Wert für Anfang ist. Die gesamte Schleife wird dann rückwärts ausgeführt. Wenn die Schleife einmal begonnen ist, dann kann Index - wie jede andere normale Variable - dem Programm entnommen werden. Sie können den Wert jedoch nicht ändern, denn das würde eine Fehlermeldung hervorrufen.

Jede FOR-Anweisung in Ihrem Programm muß von einer NEXT-Anweisung begleitet werden. Sie können dafür die in anderen Basic-Versionen erscheinenden Kurzformen wie NEXT R1,R1 allerdings nicht verwenden. Hier sind einige einfache Beispiele für diese Schleifen:

```
For I=32 To 255:Print Chr$(I);:Next I

For R1=20 To 100 Step 20
    For R2=20 To 100 Step 20
          For A=0 To 3
              Ink A
              Ellipse 160,100,R1,R2
          Next A
    Next R2
Next R1
```

Sie sehen hier, wie wir eine Reihe von FOR...NEXT Schleifen ineinander gestellt haben. Das nennt man "Verschachtelung".

WHILE...WEND

(Wiederholt einen Code-Abschnitt, wenn eine bestimmte Bedingung zutrifft)

Der Befehl WHILE bietet Ihnen eine nützliche Methode, eine Reihe von Basic-Anweisungen zu wiederholen, bis eine bestimmte Bedingung erfüllt wird.

```
WHILE Bedingung
:              :
Liste der Anweisungen
:              :

WEND
```

Die Bedingung kann dabei aus jedem beliebigen Satz von Tests bestehen und die Anweisungen AND, OR und NOT enthalten. Bei jeder Windung der Schleife wird automatisch ein Test durchgeführt. Wenn dieser Test den korrekten Wert von -1 (true) ergibt, so werden die Anweisungen zwischen WHILE und WEND ausgeführt. Andernfalls wird die Schleife abgebrochen, und Ihr Basic springt zur nächsten Anweisung. Geben Sie dazu folgendes Beispiel ein:

```
Input "Gib eine Zahl ein";X
Print "Bis 11 zählen"
While X<11
  Inc X
  Print X
Wend
Print "Schleife beendet"
```

Wie oft die WHILE-Schleife in diesem Programm ausgeführt wird hängt von dem Wert ab, den Sie für diese Routine eingeben. Wenn Sie eine Zahl über 10 eingeben, läuft sie überhaupt nicht ab. WHILE führt deshalb die Anweisungen nur aus, wenn die Bedingung am Anfang Ihres Programms auch zutrifft (TRUE).

REPEAT...UNTIL *(Wiederholen, bis eine Bedingung erfüllt wird)*

REPEAT
: :
Liste von Anweisungen
: :
UNTIL Bedingung

Der Befehl REPEAT...UNTIL ist dem Befehl WHILE...WEND sehr ähnlich, abgesehen davon, daß der Test für die Ausführung der Anweisungen am Ende der Schleife anstatt am Anfang erfolgt. Die Schleife wird ständig wiederholt, bis die angegebene Bedingung falsch (FALSE) ist. Somit wird die Schleife in Ihrem Programm also mindestens einmal durchgeführt. Hier ist ein Beispiel:

```
Repeat
 Print "AMOS Basic"
Until Mouse Key<>0
```

Wie auch beim Befehl WHILE...WEND sollten Sie nie vergessen, daß auf ein REPEAT stets ein UNTIL folgen muß.

DO...LOOP *(Endlosschleife)*

DO
: :
Liste von Anweisungen
: :
LOOP

Mit dem Befehl DO...LOOP wird eine Liste von Basic-Anweisungen ununterbrochen wiederholt. Um aus dieser Schleife wieder auszusteigen, müssen Sie einen speziellen EXIT oder EXIT IF Befehl eingeben.

```
Do
  Print "Hallo"
Loop
```

Dieses System hat den Vorteil, daß es eine Strukturalternative zu den GOTO- Schleifen ist, die in früheren Basic-Versionen gerne auftauchten. Wir wollen uns dazu einmal folgendes Beispiel ansehen:

```
TEST:
Input "Noch ein Spiel (J;N)";AN$
If Upper$(AN$)="N" Then Goto TSCHÜSS
GAME : Rem ruft Spielablauf auf
Goto Test
TSCHÜSS:
End
```

Auf eine solche Routine stößt man ziemlich häufig, sie ist jedoch nicht besonders leicht zu lesen. Jetzt wollen wir mit DO...LOOP eine zweite Version probieren:

```
Do
    Input "Noch ein Spiel (J;N)";AN$
    Exit If Upper$(AN$)="N"
    GAME : Rem ruft Spielablauf auf
Loop
End
```

Hoffentlich glauben auch Sie jetzt, daß die neue Routine viel übersichtlicher ist!

EXIT *(Aussteigen aus einem DO...Loop)*

EXIT [n]

Mit dem Befehl EXIT können Sie sofort aus einer oder mehreren Programmschleifen aussteigen, die mit den Anweisungen FOR...NEXT, REPEAT...UNTIL, WHILE...WEND oder DO...LOOP erzeugt wurden. Ihr AMOS-Programm springt jetzt direkt zu der nächsten Anweisung nach der Schleife.

n ist dabei die Anzahl der Schleifen, die nicht ausgeführt werden sollen. Wenn Sie für n keinen Wert eingeben, dann wird nur die innerste Schleife beendet. Hier ist ein Beispiel dafür:

```
Do
    Do
      Print "Innere Schleife"
      If Mouse Key=1 Then Exit
```

```
    If Mouse Key=2 Then Exit 2
    Wait 5 : Rem Langsame Schleife, damit Sie den Ablauf sehen können
  Loop
  Locate 20,: Print "Äußere Schleife"
  Wait 5
Loop
```

EXIT IF *(Das Aussteigen aus einer Schleife hängt von einem Test ab)*

EXIT IF Ausdruck[,n]

Wie Sie aus der vorhergehenden EXIT-Anweisung ersehen können, erfordert das Ergebnis bestimmter Bedingungen oft das Beenden einer Schleife. Durch einen speziellen EXIT IF Befehl ist das ganz einfach.

Der Ausdruck besteht dabei aus einer Reihe von Tests im AMOS-Standardformat. Der Befehl EXIT wird nur ausgeführt, wenn des Ergebnis -1 (TRUE) lautet.

Der Parameter n ermöglicht Ihnen den Ausstieg aus mehreren Schleifen gleichzeitig. Wenn n nicht angegeben wird, dann wird nur die aktuelle Schleife abgebrochen. Hier ist ein Beispiel:

```
While L=0
  Z=0
  Do
    Z=Z+1
    For X=0 To 100
      Exit If Z=10,2 :Rem Ausstieg aus 2 Schleifen,DO und FOR
    Next X
  Loop
  Exit 1 : Rem Beendet While...Wend
Wend
```

EDIT *(Beenden des Programmablaufs und Rückkehr zum Editor)*

EDIT

Mit dem EDIT-Befehl können Sie das gegenwärtig ablaufende Programm beenden und zum AMOS Basic-Editor zurückkehren. Das kann zum Beispiel sehr nützlich sein, wenn Sie gerade die Bugs in einem Ihrer Programme ausmerzen.

DIRECT *(Aussteigen in den Direktmodus)*

DIRECT

DIRECT beendet Ihr Programm und springt sofort in den Direktmodus zurück. Sie können jetzt den Inhalt Ihrer Variablen untersuchen oder Ihre Programm-Listings ausdrucken.

END *(Aussteigen aus dem Programm)*

END

Mit diesem Befehl können Sie aus einem Programm aussteigen. Sie erhalten danach die Option, entweder in den Editier- oder den Direktmodus zurückzukehren. Durch Drücken der Leertaste können Sie Ihr Programm bearbeiten, und mit ESCAPE springen Sie in den Direktmodus.

ON...PROC *(Springt zu einer von mehreren Prozeduren je nach Variable)*

ON v PROC proc1, proc2, proc3 ...procN

Mit diesem Befehl können Sie, abhängig vom Inhalt der Variable v, zu einer bestimmten Prozedur springen. Beachten Sie bitte, daß keine der Prozeduren, die Sie bei diesem Befehl einsetzen, Parameter enthalten dürfen. Wenn Sie in diese Prozedur Information übertragen müssen, dann sollten Sie sie stattdessen in globale Variablen setzen. Unter PROZEDUREN (PROCEDURES) in Kapitel 4 finden Sie eine genaue Erklärung dieser Technik.

Der Befehl On...Proc entspricht eigentlich den folgenden Zeilen:

```
If v=1 Then Proc1
If v=2 Then Proc2
:   :    :
If v=n Then ProcN
```

ON...GOTO *(Springt zu einer Reihe von Zeilen je nach Variable)*

ON v GOTO Zeile1, Zeile2, Zeile3...ZeileN

Die Anweisung On GOTO läßt Ihr Programm zu einer bestimmten Zeile aus einer Reihe von Zeilen springen, je nachdem, wie das Ergebnis des Ausdrucks v lautet. Sie entspricht den folgenden Zeilen:

```
If v=1 Then Goto Zeile1
```

```
If v=2 Then Goto Zeile2
:    :       :
If v=n Then Goto ZeileN
```

Damit diese Anweisung jedoch wirksam wird, muß n eine normale ganze Zahl zwischen 1 und der Anzahl aller möglichen Zielorte sein. Zeile kann entweder eine Zeilennummer oder eine Sprungmarke sein. Siehe auch die Befehle GOTO, GOSUB, ON GOSUB.

ON...GOSUB

(GOSUB zu einer von einer Reihe von Routinen, je nach Variable)

ON var GOSUB Zeile1, Zeile2, Zeile3...

Dieser Befehl ist identisch mit ON...GOTO, es wird jedoch mit einem Gosub anstelle eines Goto in die betreffende Zeile gesprungen. Wenn eine Subroutine ganz abgelaufen ist, dann sollten Sie mit einem RETURN-Befehl zur nächsten Anweisung nach dem ON...GOSUB-Befehl zurückspringen.

Siehe auch GOSUB und GOTO.

EVERY n GOSUB

(Aufrufen einer Subroutine in regelmäßigen Abständen)

EVERY n GOSUB Label

Die Anweisung ON EVERY ruft die Subroutine an der angegebenen Sprungmarke in regelmäßigen Abständen auf. Dies hat jedoch keine Auswirkungen auf Ihr Hauptprogramm.

n stellt dabei die Länge des Intervalls in Fünfzigstel einer Sekunde dar. Die Zeit, die Ihre Subroutine zum Ablaufen benötigt, muß immer unter dieser Zeitspanne liegen, sonst erhalten Sie eine Fehlermeldung.

Nachdem Sie eine Subroutine eingegeben haben, wird das System automatisch außer Kraft gesetzt. Um diese Komponente ständig aufzurufen, müssen Sie deshalb einen EVERY ON-Befehl vor der letzten RETURN-Anweisung einzufügen. Sehen wir uns das an folgendem Beispiel an:

```
Every 50 Gosub TEST
Do
   Print "Hauptschleife"
Loop
TEST:
Inc I : Print "Das ist Abruf Nummer";I
Every On:Return
```

Beachten Sie bitte, daß ON EVERY dem Befehl ON TIME in Amiga Basic sehr ähnlich ist.

EVERY n PROC

(Ruft eine Prozedur in regelmäßigen Abständen auf)

EVERY n PROC Name

EVERY PROC führt die erforderliche Prozedur automatisch in regelmäßigen Abständen mittels eines leistungsstarken Unterbrechersystems aus.

n stellt die Verzögerung, gemessen in Fünfzigstel einer Sekunde zwischen jedem Abruf der Prozedur dar.

Wie bei dem vorhergehenden Befehl muß der Unterbrecher vor dem Verlassen des Ablaufs wieder aktiviert werden, sonst wird die Routine nur einmal aufgerufen. Sie müssen die Anweisung EVERY ON also einsetzen, bevor Sie mit END PROC in Ihr Hauptprogramm zurückkehren.

```
Every 50 Proc TEST
Do
   Print "Hauptschleife"
Loop
Procedure TEST
   Shared I
   Inc I : Print "Das ist Abruf Nummer";I
   Every On
End Proc
```

EVERY ON/OFF

(Ein- und Ausschalten des automatischen Abrufs von Prozeduren)

EVERY ON/OFF

EVERY On aktiviert das Unterbrechersystem, das für die EVERY-Befehle eingesetzt wird, wieder. Diese Anweisung sollte ganz kurz vor dem Ende der Prozedur oder Subroutine aufgerufen werden.

Ganz ähnlich wird der Abruf durch den Befehl EVERY OFF völlig unterbunden. Dieser Befehl wird automatisch zu Beginn der betreffenden Prozedur ausgeführt.

BREAK ON/OFF

(Ein- und Ausschalten der Abbruchtasten Control+C)

BREAK ON/OFF

Normalerweise können Sie ein Programm jederzeit durch Drücken der Tastenkombination Ctrl+C unterbrechen und ins Basic zurückkehren. Diese Funktion kann durch den Befehl BREAK OFF außer Kraft gesetzt und Ihr Programm so mit einem groben Kopierschutz versehen werden. Wie Sie wahrscheinlich auch schon vermuteten, können Sie die Abbruchtasten mit dem Befehl BREAK ON wieder aktivieren.

Hier jedoch ein Wort der Warnung: Lassen Sie nie ein Programm mit Abbruchschutz

ablaufen, ohne vorher eine Sicherungskopie auf Diskette zu machen. Wenn sich Ihr Programm nämlich in einer Schleife aufhängt, dann können Ihnen ganz schnell einige Stunden Arbeit verlorengehen.

Wie man mit Fehlern umgeht

ON ERROR GOTO *(Erfaßt einen Fehler in einem Basic-Programm)*

ON ERROR GOTO Label

Mit dem Befehl ON ERROR können Sie in einem AMOS Basic-Programm Fehler ermitteln und korrigieren, ohne in das Editier-Fenster zurückkehren zu müssen. Manchmal können in einem Programm Fehler auftauchen, die man unmöglich vorhersehen konnte. Nehmen wir zum Beispiel einmal die folgende Routine:

```
Do
   Input "Zwei Zahlen eingeben";A,B
   Print A;"dividiert durch ";B;" ist ";A/B
Loop
```

Dieses Programm funktioniert ganz gut, bis Sie auf die Idee kommen, für B Null einzugeben. Dann versucht AMOS-Basic nämlich, A durch Null zu dividieren, und Sie erhalten die Fehlermeldung Teilung durch Null.

Dieses Problem können Sie vermeiden, indem Sie den Fehler folgendermaßen mit dem Befehl ON ERROR GOTO erfassen:

ON ERROR GOTO Label

Jetzt wird AMOS bei jedem Fehler in Ihrem Basic-Programm direkt zu dieser Sprungmarke springen. Diese Sprungmarke markiert den Anfangspunkt Ihrer eigenen Korrektur-Routine, mit der Sie den Fehler beseitigen und sicher in Ihr Hautprogramm zurückkehren können.

Beachten Sie hier bitte, daß die Korrektur-Routine mit einer speziellen RESUME-Anweisung arbeiten muß. Sie können nämlich nicht mit einem normalen GOTO- Befehl in Ihr Programm zurückspringen.

```
On Error Goto HILFE
Do
   Input "Zwei Zahlen eingeben";A,B
   Print A;"dividiert durch ";B;"ist ";A/B
Loop
Rem Korrektur-Routine
HILFE:
```

```
Print : Print "Alarm"
Print "Ich fürchte du hast versucht"
Print "deine Zahl durch Null zu dividieren."
Resume Next: Rem Zurück zur nächsten Anweisung
```

Damit dieses System funktioniert, darf in Ihrer Korrektur-Routine kein Fehler auftauchen, sonst bringt AMOS Ihr Programm unbarmherzig zum Stillstand.

Die Ausführung des Befehls ON ERROR GOTO kann durch Aufrufen von ON ERROR ohne Angabe von Parametern unterbunden werden.

```
On Error : Rem Klappe zu - Korrektur vorbei
```

Sie können zu diesem Zweck auch ON ERROR GOTO 0 einsetzen.

ON ERROR PROC *(Korrigiert einen Fehler mit einer Prozedur)*

ON ERROR PROC Name

Sie können damit eine Prozedur wählen, die automatisch aufgerufen wird, sobald ein Fehler im Hauptprogramm auftaucht. Hier handelt es sich eigentlich nur um eine strukturierte Ausführung der oben erklärten Anweisung ON ERROR GOTO.

Obwohl Ihre Prozedur mit dem üblichen END PROC abgeschlossen werden muß, können Sie nur in Ihr Hauptprogramm zurückkehren, wenn Sie außerdem RESUME aufrufen. Diese Anweisung kann unmittelbar vor dem letzten END PROC Befehl eingeführt werden. Hier ist ein Beispiel dafür:

```
On Error Proc HILFE
Do
   Input "Zwei Zahlen eingeben";A,B
   Print A;"dividiert durch ",B;" ist ";A/B
Loop
Rem Fehlerkorrektur
Procedure HILFE
   Print : Print "Alarm"
   Print "Ich fürchte, du hast versucht"
   Print "deine Zahl durch Null zu dividieren."
Resume Next: Rem Von der nächsten Anweisung aus ins Basic zurück
End Proc
```

Ihre Korrektur-Routine kann jede beliebige Basic-Prozedur aufrufen. Darüber hinaus können Sie jede Routine mit ihrem eigenen Fehlerschutz versehen. Fehler in der Korrektur-Routine selbst können jedoch nicht ermittelt werden. Wenn AMOS also auf ein Problem in der HILFE-Routine stößt, wird Ihr Programm unverzüglich abgebrochen.

RESUME *(Wiederaufnehmen des Programms nach einem Fehler)*

Mit RESUME können Sie zu einem Basic-Abschnitt zurückkehren, nachdem die Korrektur-Routine, die Sie mit dem Befehl ON ERROR erstellten, das ursprüngliche Problem beseitigt hat. Sie sollten NIEMALS versuchen, den Befehl GOTO in diesem Zusammenhang anzuwenden.

Für diese Anweisung gibt es fünf verschiedene Formate:

RESUME

Springt zu der Anweisung zurück, die die Fehlermeldung hervorrief, und versucht sie nochmals auszuführen.

RESUME NEXT

Kehrt zu dem Befehl nach der Anweisung zurück, die den Fehler generierte.

RESUME Zeile

Springt an einen bestimmten Punkt in Ihrem Hauptprogramm zurück. Zeile kann dabei entweder eine Sprungmarke oder eine normale Zeilennummer darstellen. Mit diesem Befehl können Sie NICHT wieder in eine Prozedur einsteigen.

Prozeduren werden etwas anders gehandhabt. Wenn Sie zu einer bestimmten Sprungmarke springen wollen, dann müssen Sie in der Prozedur, die Sie auf Fehler untersuchen, eine besondere Markierung anbringen. Dafür können Sie den Befehl RESUME LABEL einsetzen, für den es zwei verschiedene Versionen gibt:

RESUME LABEL Label

Definiert die Sprungmarke, zu dem Sie nach dem Fehler zurückkehren wollen. Diese Anweisung muß außerhalb Ihrer Korrektur-Routine unmittelbar nach dem ursprünglichen ON ERROR PROC oder ON ERROR GOTO Befehl aufgerufen werden.

RESUME LABEL

Diese Anweisung können Sie innerhalb ihrer Korrektur-Routine einsetzen, um direkt zu der Sprungmarke, das Sie mit dem vorhergehenden Befehl markiert haben, zurückzukehren. Hier ist ein Beispiel:

```
On Error Proc HILFE
Resume Label NACH
Error 12
Print "Nie gedruckt"
```

```
NACH: Print "Ich bin hierher zurück"
End
Procedure HILFE
   Print "Oh Schreck, ich glaub' da ist ein Fehler!"
   Resume Label
Endproc
```

=ERRN *(Gibt die Nummer des letzten Fehlers an)*

e=ERRN

Wenn Sie Ihre eigene Korrektur-Routine mit dem Befehl ON ERROR erstellen, dann müssen Sie genau feststellen können, welcher Fehler in Ihrem Hauptprogramm aufgetreten ist. Wenn ein Fehler auftaucht, wird ERRN automatisch mit seiner Identifikationsnummer geladen. In Anhang am Ende dieser Anleitung finden Sie eine vollständige Liste aller potentiellen Fehler.

```
Print Errn
```

ERROR *(Ruft eine Fehlermeldung hervor und kehrt in den AMOS Editor zurück)*

ERROR n

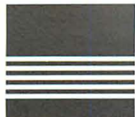
Der Befehl ERROR ruft im allgemeinen einen Fehler hervor. Das klingt vielleicht etwas verrückt, kann aber manchmal ganz nützlich sein. Nehmen wir einmal an, Sie haben gerade eine nette, kleine Korrektur-Routine erstellt, die alle möglichen Diskettenfehler behandelt. Dann können Sie mit ERROR jetzt die verschiedenen Probleme simulieren, ohne einen tatsächlichen Fehler in Kauf nehmen zu müssen. Zum Beispiel:

```
Error 40
```

Verläßt das Programm und druckt die Fehlermeldung Sprungmarke nicht definiert aus. Eine andere sehr nützliche Anwendung dieses Befehls ist:

```
Error Errn
```

Hier wird die ERRN-Funktion dazu eingesetzt, die aktuelle Fehlerbedingung nach einem Problem in Ihrem Programm anzuzeigen.



8: Text und Fenster

In diesem Kapitel behandeln wir die Optionen für Text und Fenster, die AMOS-Basic unterstützt. Es stehen Ihnen unzählige Befehle zur Verfügung, mit welchen Sie so ziemlich alles von einfachen Tabellen zur Darstellung des Spielstandes bis hin zu superduper Dialog-Boxen erzeugen können.

Text-Attribute

PEN *(Wahl der Farbe des Textes)*

PEN Index

Mit der Anweisung PEN können Sie die Farbe, in der der Text im aktuellen Fenster dargestellt wird, auswählen. Für diese Farbe stehen bis zu 64 verschiedene Möglichkeiten zur Wahl, je nachdem, in welchem Grafik-Modus Sie gerade arbeiten. Hier ist ein Beispiel:

```
For INDEX=0 To 15
Pen INDEX
   Print "Stift Nummer";INDEX;At(20,);"Farbe"
Next INDEX
```

Als Default ist die Farbe Nummer 2 (weiß) eingestellt.

=PEN$ *(Wechseln der Farbe durch Steuerzeichen)*

a$=PEN$(n)

Durch PEN$ erhält man eine besondere Kontroll-Sequenz, die die Farbe in einer Zeichenkette (String) verändert. Die neue Farbe wird automatisch zugewiesen, sobald die Zeichenkette danach auf dem Bildschirm erscheint. Dazu folgendes Beispiel:

```
C$=Pen$(2)+"Weiß"+Pen$(6)+"Blau"
Print C$
```

Die Zeichenkette, die von PEN$ ausgegeben wird, hat das Format: Chr$(27)+"P"+Chr$(48+n).

Siehe dazu auch die Befehle COLOUR, PALETTE und PAPER.

PAPER *(Wahl der Farbe für den Texthintergrund)*

PAPER Index

Mit dem Befehl PAPER können Sie eine Farbe für den Hintergrund Ihres Textes wählen. Wie bei PEN können Sie unter den Farbnummern 0 bis 63 auswählen. Zum Beispiel:

```
Pen 2 : For INDEX=0 To 15
   Paper INDEX : Print "Papier Nummer";INDEX;Space$(10)
Next INDEX
```

Der Default für den Bildschirmhintergrund ist normalerweise auf Farbe Nummer Eins (orange) gestellt. Unter SCREEN OPEN werden die verschiedenen anderen Möglichkeiten aufgeführt.

=PAPER$ *(Gibt eine Kontroll-Sequenz zur Wahl der Hintergrundfarbe an)*

x$=PAPER$(Index)

Der Befehl PAPER$ liefert eine besondere Kontroll-Sequenz, die automatisch die Farbe des Hintergrunds ändert, wenn sie auf dem Bildschirm erscheint. Hierzu ein Beispiel:

```
Pen 1 : C$=Paper$(2)+"Weiß"+Paper$(6)+"Blau"
Print C$
```

Siehe auch die Befehle PEN, COLOUR und PALETTE.

INVERSE ON/OFF *(Aufrufen des inversen Modus)*

INVERSE ON
INVERSE OFF

Mit dem Befehl INVERSE können Sie die Farben für Vorder- und Hintergrund, die Sie mit den Befehlen PEN und PAPER ausgewählt haben, vertauschen. Diese Anweisung wirkt nur auf den Text, der im aktuellen Fenster dargestellt ist.

INVERSE ON aktiviert diesen Modus, und mit INVERSE OFF wird er folgerichtig wieder abgestellt. Hier ein Beispiel:

```
Print "Das ist Text im normalen Modus"
Inverse On : Print "So sieht inverser Text aus":Inverse Off
```

Siehe hierzu auch die Befehle SHADE, UNDER und WRITING.

SHADE ON/OFF *(Hebt den nachfolgenden Text hervor)*

SHADE ON
SHADE OFF

Mit dem Befehl SHADE ON können Sie Ihren Text hervorheben, denn die Leuchtkraft der Zeichen wird durch ein Maskenmuster vermindert. SHADE OFF macht die Hervorhebung rückgängig, und der Text wird wieder ganz normal dargestellt. Dafür ein Beispiel:

Shade On : Print "Hervorgehobener Text"
Shade Off : Print "Normaler Text"

Siehe auch die Befehle UNDER, INVERSE und WRITING.

UNDER ON/OFF *(Unterstreichen)*

UNDER ON
UNDER OFF

Nach der Eingabe von UNDER ON wird Ihr Text auf dem Bildschirm unterstrichen dargestellt; mit UNDER OFF können Sie das Unterstreichen wieder abstellen. Zum Beispiel:

Under On :Print "Unterstrichen"
Unterstrichen
Under Off:Print "Normal"
Normal

Siehe dazu auch die Befehle SHADE, INVERSE und WRITING.

WRITING *(Verändert den Modus für das Schreiben von Text)*

WRITING w1 [,w2]

Mit dem Befehl WRITING können Sie den Schreib-Modus für alle folgenden Textoperationen verändern. Sie bestimmen damit genau, wie Ihr neuer Text mit den bereits vorhandenen Daten auf dem Bildschirm verbunden wird.

Der erste Wert bezeichnet einen der vier Schreib-Modi:

w1=0	REPLACE (Default)	Ihr Text überschreibt alles Darunterliegende.
w1=1	OR	Verbindet die Zeichen auf dem Bildschirm mit einem logischen OR.
w1=2	XOR	Die Zeichen werden hier durch XOR mit den Daten auf dem Bildschirm verknüpft.
w1=3	AND	Der neue Text wird durch AND mit dem Hintergrund verbunden.
w1=4	IGNORE	Alle Druckoperationen werden jetzt völlig ignoriert.

Der zweite Wert dient zur Auswahl der Textabschnitte, die auf dem Bildschirm angezeigt werden sollen. Diese Option kann - falls erforderlich - auch weggelassen werden.

w2=0	Normal	Der Text wird auf dem Bildschirm zusammen mit seinem Hintergrund dargestellt.
w2=1	Paper	Nur der Hintergrund erscheint auf dem Bildschirm.
w2=2	Pen	Ignoriert die für den Hintergrund gewählte Farbe und stellt den Text auf einem Hintergrund der Farbe Null dar.

Verwechseln Sie diesen Befehl bitte nicht mit GR WRITING.

Cursor-Funktionen

Wenn Sie die PRINT-Anweisung einsetzen, dann werden Ihre Zeichen immer an der aktuellen Cursor-Position dargestellt. AMOS verfügt jedoch auch über eine Reihe von Möglichkeiten, durch die Sie den Cursor zu jedem beliebigen Punkt auf dem Bildschirm bewegen können.

LOCATE *(Positionieren des Cursors)*

LOCATE x,y
LOCATE x,
LOCATE ,y

Der Befehl LOCATE bewegt den Cursor zu den Koordinaten x,y. Dort liegt dann der Ausgangspunkt für alle weiteren Druckoperationen. Alle Positionen auf dem Bildschirm werden durch einen besonderen Satz von Textkoordinaten bestimmt. Die Maßeinheit der Koordinaten stellt die Entfernung eines Zeichens von der linken oberen Ecke des Textbildschirms dar. Um zum Beispiel zu den Koordinaten 15,10 zu gelangen, muß man 10 Zeichen nach unten und 15 Zeichen nach rechts gehen.

Der für diese Koordinaten akzeptable Bereich ist unterschiedlich und hängt von den jeweiligen Dimensionen Ihres Fensters und der Größe Ihres Zeichensatzes ab. Wenn Sie versuchen, außerhalb dieser Begrenzungen etwas zu drucken, erhalten Sie eine Fehlermeldung.

Beachten Sie auch, daß der aktuelle Schirm immer als Fenster 0 gilt. Sie müssen also nicht erst ein Fenster eröffnen, um eine dieser Funktionen nutzen zu können. Dazu ein Beispiel:

```
Locate 10,10 : Print "Hallo"
```

Wenn Sie jetzt den Cursor auf der aktuellen Zeile positionieren wollen, dann können Sie die Y-Koordinate ganz weglassen. Zum Beispiel:

```
Print "Höchststand 10000"; : Locate 9, : Print "12345";
```

Sie können den Cursor auch ohne Auswirkung auf die vorhandene x-Koordinate vertikal verschieben.

```
Clw : Locate ,10 : Print "Zehnte Zeile";
```

Siehe auch die Befehle CMOVE, AT, XCURS und YCURS.

CMOVE *(Relative Cursor-Bewegung)*

CMOVE w,h

CMOVE bewegt den Cursor zu einer neuen Position in einer bestimmten Entfernung von seinem aktuellen Standort. Dabei wird der Inhalt der Variablen w und h zu den gegenwärtigen Cursor-Koordinaten addiert. Wenn der Cursor also auf 10,10 stand, und man dann:

```
Cmove 5,-5
```

eingibt, bewegt sich der Cursor zu den Koordinaten 15,5.

Wie bei dem Befehl LOCATE können Sie auch wieder eine der beiden Koordinaten weglassen. Das sieht dann zum Beispiel so aus:

```
Cmove ,2 : Rem Bewegt den Cursor zwei Stellen nach unten
Cmove 2, : Rem.Bewegt den Cursor zwei Stellen nach rechts
```

=AT *(Gibt eine Kontroll-Sequenz zur Positionierung des Cursorsan)*

X$=AT(x,y)

Die AT-Funktion ermöglicht es Ihnen, die Position des Textes direkt aus der Zeichenkette

heraus zu verändern. Dies funktioniert durch die Darstellung der Sequenz in folgendem Format:

```
Chr$(27)+"X"+Chr$(48+X)+Chr$(27)+"Y"+Chr$(48+Y)
```

Wenn diese Sequenz erscheint, springt der Cursor zu den Koordinaten x,y. Dazu folgendes Beispiel:

```
A$="Das"+At(10,10)+"schafft"+At(1,2)+"nur"+At(20,20)+"AMOS!"
Print A$
```

Solche AT-Befehle eignen sich ganz besonders gut für die Erstellung von Punktestandtabellen, weil Sie so Ihren Text während der Initialisierungsphase Ihres Programms ein für allemal positionieren können. Sie können dann das Punktergebnis an der richtigen Stelle auf dem Bildschirm mit einem einzigen Print- Befehl auf den neuesten Stand bringen. Hier ist ein Beispiel dafür:

```
PUNKT_STAND$=At(20,10)+"Punktstand"
STAND=1000
Print PUNKT_STAND$;STAND
```

Das stimmt genau mit den Zeilen:

```
STAND=1000
Locate 20,10 : Print "Punktstand";STAND
```

überein. Die erstere Fassung ist allerdings leichter zu ändern, weil Sie die Tabelle nur durch Überarbeiten einer Sequenz verschieben können. Dabei spielt es keine Rolle, wie oft Sie sie in Ihrem Programm einsetzen.

Siehe auch LOCATE, CMOVE, CUP$, CDOWN$, CLEFT$ und CRIGHT$.

Konvertierungs-Funktionen

AMOS-Basic bietet Ihnen vier nützliche Funktionen, die die Konvertierung von Text- und Grafikkoordinaten ganz leicht machen.

=XTEXT *(Konvertiert eine X-Koordinate aus dem Grafik- in das Textformat)*

t=XTEXT(x)

Diese Funktion erfaßt eine normale X-Koordinate und konvertiert sie in eine Text-Koordinate mit Bezug zum aktuellen Fenster. Wenn die Bildschirm-Koordinate außerhalb dieses Fensters liegt, so wird sie als negativer Wert dargestellt. Siehe **BEISPIEL 8.1**. Siehe auch die Befehle YTEXT, LOCATE, WINDOPEN, XGRAPHIC und YGRAPHIC.

YTEXT *(Konvertiert eine Y-Koordinate aus dem Grafik- in das Textformat)*

t=YTEXT(y)

Der Befehl YTEXT konvertiert eine y-Koordinate des Standard-Bildschirmformats in eine Text-Koordinate mit Bezug zum aktuellen Fenster.

Nähere Einzelheiten dazu siehe unter XTEXT. Siehe auch YGRAPHIC, XGRAPHIC und LOCATE.

=XGRAPHIC *(Konvertiert eine x-Koordinate aus dem Text- in das Grafikformat)*

g=XGRAPHIC(x)

Die Funktion XGRAPHIC ist praktisch die Umkehrung von XTEXT, da hier eine x- Text-Koordinate aus dem Bereich von O bis zur Gesamtbreite des aktuellen Fensters genommen und in eine absolute Bildschirm-Koordinate umgesetzt wird. Mit dieser Funktion können Sie Text über einen Grafikbereich auf dem Bildschirm setzen. In **BEISPIEL 8.2** des MANUAL-Unterverzeichnisses erhalten Sie eine Demonstration dieser Funktion. Siehe dazu auch die Befehle YGRAPHIC, XTEXT und YTEXT.

=YGRAPHIC *(Konvertiert eine y-Koordinate aus dem Text- in das Grafikformat)*

g=YGRAPHIC(y)

Diese Funktion konvertiert eine y-Text-Koordinate in eine absolute Bildschirm-Koordinate.

Siehe XGRAPHIC, XTEXT und YTEXT.

Cursor-Befehle

Der Text-Cursor dient als sichtbarer Ausgangspunkt für alle weiteren Textoperationen. Er wird normalerweise als blinkender, horizontaler Balken dargestellt, dies kann aber durch die Befehle SET CURS und CURS OFF verändert werden.

Durch Verschieben des Cursors auf dem Bildschirm können Sie Ihren Text praktisch beliebig positionieren. Vergessen Sie aber nicht, daß alle Koordinaten-Angaben auf den Text-Koordinaten in Relation zum aktuellen Fenster beruhen.

AMOS bietet Ihnen unzählige einfache Befehle, durch die Sie den Cursor ganz exakt auf dem Bildschirm positionieren können. Sie können auch direkt von Ihrem Basic-Programm aus das Erscheinungsbild des Text-Cursors verändern.

HOME *(Cursor an den Ausgangspunkt)*

HOME

Der Befehl HOME bringt den Cursor an den Ausgangspunkt in die linke obere Ecke des aktuellen Fensters (Koordinaten 0,0). Zum Beispiel:

```
Clw : Rem Löscht aktuelles Fenster
Locate 10,10 : Rem Bringt Cursor zu 10,10
Print "Eine Demonstration von"
Home : Print "HOME" : Rem Bringt Cursor zu 0,0
```

Siehe auch die Befehle LOCATE, XCURS und YCURS.

CDOWN *(Cursor nach unten)*

CDOWN

Mit dem Befehl CDOWN schiebt man den Text-Cursor eine Zeile nach unten. Hier ist ein Beispiel dafür:

```
Print "Beispiel" : Cdown : Cdown : Print "von Cdown"
```

=CDOWN$ *(Gibt ein chr$(31)-Zeichen an)*

x$=CDOWN$

CDOWN$ ist eine Funktion, die ein spezielles Steuerzeichen liefert, das den Cursor automatisch verschiebt, sobald es auf dem Bildschirm erscheint. So stimmt also Print CDOWN$ mit CDOWN genau überein. Mit CDOWN$ können Sie jedoch mehrere Cursor-Bewegungen in einer Zeichenkette (String) verbinden. Das kann manchmal äußerst nützlich sein. Dazu folgendes Beispiel:

```
C$="\"+Cdown$
For A=0 to 20
   Print C$;
Next A
```

Siehe auch CUP, CLEFT, CRIGHT und AT.

CUP *(Cursor nach oben)*

CUP

Mit dem Befehl CUP kann man den Cursor eine Zeile nach oben bewegen wie mit CDOWN nach unten. Zum Beispiel:

Print "Beispiel" : Cup : Cup : Print "für cup"

=CUP$ *(Ergibt ein chr$(30)-Zeichen)*

X$=CUP$

Durch CUP$ erhält man eine Kontroll-Sequenz, die den Cursor ein Zeichen nach oben bewegt. Das sieht zum Beispiel so aus:

Print "Der Cursor springt"+CUP$+"eine Zeile rauf..."

Siehe auch CLEFT, CDOWN, CRIGHT und AT.

CLEFT *(Cursor nach links)*

CLEFT

Die Anweisung CLEFT schiebt den Cursor ein Zeichen nach links. Zum Beispiel:

Print "Beispiel" : Cleft : Cleft : Print "für cleft"

=CLEFT$ *(Generiert ein chr$(29)-Steuerzeichen für CLEFT)*

x$=CLEFT$

Die Funktion CLEFT$ liefert ein Steuerzeichen, das einen CLEFT-Befehl ausführt. Ein Beispiel hierfür:

Print "Hallo";
Print Cleft$+Cleft$+"p";

Siehe auch die Befehle CUP, CRIGHT, CDOWN, AT.

CRIGHT *(Cursor nach rechts)*

CRIGHT

CRIGHT ist genau das Gegenteil von CLEFT und bewegt den Cursor ein Zeichen nach rechts.

Print "Beispiel" : Cright : Cright : Print "für cright"

=CRIGHT$ *(Generiert ein chr$(28)-Steuerzeichen für CRIGHT)*

x$=CRIGHT$

Die Funktion CRIGHT$ liefert eine Kontroll-Sequenz, durch die in einer Textfolge ein CRIGHT-Schritt ausgeführt wird. Dazu ein Beispiel:

Print Cright$:Rem Das hat dieselbe Wirkung wie CRIGHT

Siehe auch CLEFT, CUP, CDOWN und AT.

XCURS *Gibt die x-Koordinate des Text-Cursors an)*

x=XCURS

XCURS ist eine Variable, die die aktuelle x-Koordinate des Text-Cursors (im Textformat) enthält. Hier folgendes Beispiel:

Locate 10,0 : Print Xcurs
10

YCURS *(Gibt die y-Koordinate des Cursors an)*

y=YCURS

YCURS enthält die y-Koordinate des Text-Cursors (im Textformat).

SET CURS *(Bestimmt die Form des Cursors)*

SET CURS L1,L2,L3,L4,L5,L6,L7,L8

Mit dieser Anweisung können Sie das Erscheinungsbild des Cursors beliebig verändern. Die Form des Cursors wird durch ein Reihe von Bit-Mustern bestimmt, die in den

Parametern L1-L8 untergebracht sind. Jeder Parameter bestimmt das Erscheinungsbild einer horizontalen Cursor-Zeile. Diese Zeilen sind von oben nach unten durchnumeriert.

Jedes Bit stellt einen Punkt in der aktuellen Cursor-Zeile dar. Wenn es auf den Wert "1" gestellt ist, dann wird der Punkt in der Farbe 3 erstellt, ansonsten wird er in der Farbe, die gegenwärtig für den Hintergrund gewählt ist, angezeigt. Am besten machen Sie sich anhand eines Beispiels mit dieser Anweisung vertraut.

```
L1=%11111111
L2=%11111110
L3=%11111100
L4=%11111000
L5=%11110000
L6=%11100000
L7=%11000000
L8=%10000000
Set Curs L1,L2,L3,L4,L5,L6,L7,L8
```

Normalerweise blinkt der Text-Cursor ständig. Das können Sie abstellen, indem Sie einfach den Befehl FLASH OFF aufrufen, bevor Sie diese Anweisung eingeben.

CURS ON/OFF

(Text-Cursor an-/abschalten)

CURS ON
CURS OFF

Dieser Befehl versteckt den blinkenden Cursor des aktuellen Fensters bzw. zeigt ihn wieder an. Er hat auf die Darstellung des Cursors in allen anderen Fenstern keine Auswirkung.

MEMORIZE X/Y

(Speichern der x- oder y-Koordinate des Text-Cursors)

MEMORIZE X
MEMORIZE Y

Durch den Befehl MEMORIZE wird die aktuelle Cursor-Position sicher gespeichert. Sie können jetzt jeden beliebigen Text auf den Bildschirm drucken, ohne die ursprünglichen Cursor-Koordinaten zu verlieren. Mit den REMEMBER-Befehlen können diese dann wieder geladen werden.

REMEMBER X/Y

(Abrufen der gespeicherten x- oder y-Koordinate des Text- Cursors)

REMEMBER X
REMEMBER Y

REMEMBER positioniert den Cursor auf den zuvor mit MEMORIZE gespeicherten

Koordinaten. Wenn MEMORIZE nicht eingegeben wurde, dann wird die entsprechende Koordinate auf Null gesetzt. Im MANUAL-Unterverzeichnis finden Sie unter **BEISPIEL 8.3** ein Beispiel für die Anwendung dieses Befehls.

CLINE

Löscht die aktuelle Cursor-Zeile ganz oder teilweise)

CLINE [n]

Löscht die Zeile, auf der sich der Cursor befindet. Wenn Sie für n einen Wert angegeben, dann werden n Zeichen von der aktuellen Cursor-Position an gelöscht (der Cursor wird dabei nicht verschoben).

Tippen Sie die folgenden Zeilen im Direktmodus-Fenster ein:

```
Print "Testing Testing Testing";
Cmove -7,
Cline 7
Cline
```

CURS PEN

(Wählt eine neue Farbe für den Text-Cursor)

CURS PEN n

Vertauscht die Farbe des Text-Cursors mit der Farbe, die die Index-Nummer n repräsentiert. Wenn Ihr Bildschirm-Modus Ihnen mehr als vier verschiedene Farben bietet, dann wird die Farbe 3 als Cursor-Default eingesetzt. Diese Farbe wird mit einer Blink-Sequenz animiert, die beim Laden von AMOS automatisch zugewiesen wird. Wenn Sie also eine andere Farbe wählen, dann hört der Cursor auf zu blinken. Um dann wieder einen blinkenden Cursor zu erhalten, müssen Sie mit dem FLASH- Befehl eine neue Farbsequenz definieren. Beachten Sie bitte auch, daß die neue Farbe nur im gegenwärtig geöffneten Fenster eingesetzt wird. Auf die Darstellung des Cursors in allen anderen Fenstern hat sie keinen Einfluß. Hier ist ein Beispiel dafür:

```
Curs Pen 5
```

Siehe auch die Befehle FLASH und CURS ON/CURS OFF.

Ein- und Ausgabe von Text

CENTRE *(Zentriert eine Textzeile auf dem Bildschirm)*

CENTRE a$
CENTRE erfaßt die Zeichenkette in a$ und zeigt sie in der Mitte des Bildschirms an. Der Text wird immer in der aktuellen Cursor-Zeile dargestellt. Zum Beispiel:

```
Locate 0,1
Centre "Das ist ein zentrierter TITEL"
Cmove ,3
Centre "Und hier ist noch einer"
```

=TAB$ *(Zeigt den Tabulator an)*

x$=TAB$

TAB$ zeigt ein als TAB (Ascii 9) bekanntes Kontroll-Zeichen. Wenn dieses Zeichen auf dem Bildschirm erscheint, wird der Text-Cursor automatisch einige Zeichen nach rechts geschoben. Die Größe dieses Abstandes kann mit dem Befehl SET TAB eingestellt werden. Der Default für den Tabulator-Abstand ist auf vier gestellt.

SET TAB *(Verändert den Tabulator-Abstand)*

SET TAB n

Mit diesem Befehl geben Sie den Abstand ein, um den sich der Cursor verschiebt, wenn das nächste TAB-Zeichen angezeigt wird. Hier ist ein Beispiel:

```
Home: Rem Bewegt Cursor zu den Koordinaten 0,0
Set Tab 5 : Rem Setzt Tabulator-Abstand auf 5
Print Tab$;"Hallo";: Rem Druckt Hallo angefangen bei 5,0
A$=Tab$+Tab$
Print A$;"Du" : Rem Druckt Text auf 15,0
```

Siehe auch TAB$ und CRIGHT.

REPEAT$ *(Wiederholt eine Zeichenkette)*

x$=REPEAT$(a$,n)

Die REPEAT-Funktion ermöglicht Ihnen den Ausdruck einer Zeichenkette mehrmals hintereinander mit einer einzigen PRINT-Anweisung. Das funktioniert durch Hinzufügen

einer Sequenz von Steuerzeichen zur Variablen X$. Wenn diese Sequenz dann angezeigt wird, wiederholt AMOS einfach a$ n-mal auf dem Bildschirm. Für n kann jeder beliebige Wert zwischen 1 und 207 eingegeben werden. Sie finden eine umfassende Demonstration dieses Befehls in BEISPIEL 8.4. Das Format der Kontroll-Sequenz lautet folgendermaßen:

```
Chr$(27)+"RO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Text-Befehle für Fortgeschrittene

ZONE$ *(Errichtet eine Zone um einen Textteil)*

x$=ZONE$(a$,n)

Die Funktion ZONE$ umgibt einen Textabschnitt mit einer Bildschirmzone. Nachdem Sie eine solche Zone definiert haben, können Sie mit der ZONE-Funktion überprüfen, ob es zu Kollisionen zwischen der Maus und dieser Zone kommt. Auf diese Art können Sie starke Schirm-Menüs und Dialogboxen schaffen, ohne auf irgendwelche komplizierten Programmierertricks zurückgreifen zu müssen.

a$ ist eine Zeichenkette, die den Text für eine der "Tasten" in Ihrer Dialogbox enthält. Wenn Sie x$ auf den Bildschirm schreiben, wird diese Taste automatisch aktiviert.

n gibt die Nummer der Bildschirmzone an, die Sie definieren. Wieviele solcher Zonen Sie maximal erstellen können, das hängt von dem Wert an, den Sie zuvor mit dem Befehl RESERVE ZONE festgelegt haben.

Das Programm **BEISPIEL 8.5** im MANUAL-Unterverzeichnis demonstriert den Einsatz dieses Befehls. Das Format der Kontroll-Sequenz sieht folgendermaßen aus:

```
Chr$(27)+"ZO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Siehe dazu auch die Befehle ZONE, SET ZONE, RESERVE ZONE, RESET ZONE und BORDER$.

BORDER$ *(Umrahmt einen Textteil)*

x$=BORDER$(a$,n)

Durch diesen Befehl erhalten Sie eine Folge von Steuerzeichen, die AMOS die Anweisung gibt, den gewünschten Textteil einzurahmen. Diese Anweisung wird normalerweise in Verbindung mit dem ZONE$-Befehl eingesetzt, um die interessanten Tasten zu produzieren, die man oft in Dialogboxen und Alarm- Fenstern antrifft.

n ist die Nummer der Umrandung, sie muß zwischen 1 und 16 liegen. a$ enthält den Text, der umrandet werden soll. Der Text von a$ beginnt bei der aktuellen Cursor-

Position, deshalb wundern Sie sich also nicht über ein eigenartiges Ergebnis, wenn Sie bei 0,0 mit der Anzeige beginnen. Sehen wir uns ein Beispiel an:

```
Locate 1,1 : Print Border$("AMOS Basic",1)
```

Eine eingerahmte Bildschirmzone können Sie mit einer solchen Anweisung hervorrufen:

```
Print Border$(Zone$("HIER ANKLICKEN",1),2)
```

Damit wird der Text in Zone Nummer 1 gestellt und mit Umrandung Nummer 2 eingefaßt. Als Kontroll-Sequenz wird dafür angegeben:

```
Chr$(27)+"EO"+A$+Chr$(27)+"R"+Chr$(48+n)
```

Siehe auch ZONE$, ZONE und BORDER.

HSCROLL *(Horizontales Scrollen des Textes)*

HSCROLL n

Mit diesem Befehl wird der gesamte Text in dem derzeit geöffneten Fenster, horizontal ein Zeichen nach dem anderen gescrollt. Für n können dabei die folgenden Werte eingesetzt werden:

1=Bewegt die aktuelle Zeile nach links
2=Scrollt den gesamten Bildschirm nach links
3=Bewegt die aktuelle Zeile nach rechts
4=Bewegt den Bildschirm nach rechts

Verwechseln Sie diesen Befehl nicht mit der SCROLL-Anweisung, durch die der ganze Bildschirm bewegt wird.

VSCROLL *(Vertikales Scrollen des Textes)*

VSCROLL n

Der Text im offenen Fenster wird vertikal um ein Zeichen gescrollt.

1=Der Text in der Cursor-Zeile und darunter wird nach unten gescrollt.
2=Der Text in der Cursor-Zeile und darunter wird nach oben bewegt.
3=Nur der Text über der Cursor-Zeile wird nach oben gescrollt.
4=Der Text über der Cursor-Zeile wird nach unten gescrollt.

Es werden Leerzeilen eingefügt, um die Lücke zu schließen, die das Scrollen hinterläßt.

Fenstertechnik

AMOS verfügt über eine Fenstertechnik, die es Ihnen ermöglicht, Ihre Arbeit mit Text und Grafik auf einen Teil des Bildschirms zu beschränken. Die Fenster-Befehle können in Verbindung mit den Zonen-Befehlen eingesetzt werden, um so wirkungsvolle Dialogboxen wie Auswahlboxen und Punktstandtabellen zu gestalten. Eine typische Alarmbox kann zum Beispiel leicht durch ein paar Zeilen AMOS-Basic erstellt werden.

WINDOPEN *Schafft ein Fenster)*

WINDOPEN n,x,y,w,h [,Umrandung[,Set]]

Die Anweisung WINDOPEN öffnet ein Fenster, das dann auf dem Amiga-Bildschirm angezeigt wird. In diesem Fenster werden dann alle folgenden Text-Operationen ausgeführt.

n ist die Nummer des zu definierenden Fensters. Mit AMOS können Sie beliebig viele Fenster erzeugen, die einzige Beschränkung besteht im verfügbaren Speicherplatz. Dem aktuellen Schirm wird die Fensternummer Null als Default zugewiesen. Also versuchen Sie nicht, dieses Fenster mit WINDOPEN nochmals zu eröffnen oder es mit WIND SIZE oder WIND MOVE zu verändern.

x und y sind die Grafikkoordinaten der linken oberen Ecke Ihres neuen Fensters. Da die AMOS-Fenster unter Einsatz des AMIGA Blitter-Chips gezeichnet werden, muß der Fensterbereich immer in einer 16-Pixel-Grenze liegen. Um das zu erreichen, werden die x-Koordinaten immer automatisch zum nächstliegenden Vielfachen von 16 auf- bwz. abgerundet. Wenn Sie einen Rahmen für Ihr Fenster miteinbezogen haben, dann wird die x-Koordinate außerdem noch um acht erhöht. Das stellt sicher, daß der Arbeitsbereich Ihres Fensters immer bei der richtigen Bildschirm- Grenze beginnt. Die y-Koordinate unterliegt keinerlei Beschränkungen.

w und h geben die Größe des neuen Fensters in Zeichen an. Diese Abmessungen müssen immer durch 2 teilbar sein.

Mit Umrandung können Sie den Stil der Umrandung Ihres Fensters wählen. Sie haben die Auswahl zwischen 16 verschiedenen Gestaltungsmöglichkeiten; sie entsprechen den Werten zwischen 1 und 16. Die Umrandung kann außerdem wahlweise bis zu zwei Titelzeilen aufweisen. Ein Titel wird dann oben am Fenster dargestellt, und der andere kann am unteren Rand angebracht werden.

Die AMOS-Fenster können - wie das Intuition-System - entweder Text oder Grafik enthalten. Jedem Fenster kann mit dem leistungsstarken Befehl WINDOW FONT ein eigener Zeichensatz zugewiesen werden. Darüberhinaus gibt es eine starke WIND SAVE Anweisung, die den Schirmbereich Ihres Fensters speichert. Wenn Sie eines dieser Fenster bewegen, wird der darunterliegende Inhalt automatisch neu gezeichnet. Dazu folgendes Beispiel:

```
For W=1 To 3
   Windopen W,(W-1)*96,50,10,10,1
   Paper W+3 : Pen W+6 :Clw
   Print "Fenster";W
Next W
```

Mit dem Befehl WINDOW können Sie zwischen diesen Fenstern hin- und herspringen. Versuchen Sie jetzt, die folgenden Anweisungen aus Direktmodus einzugeben:

```
Window 1 : Print "AMOS"
Window 3 : Print "in Aktion!"
Window 2 : Print "Basic"
```

Das aktive Fenster können Sie immer leicht am blinkenden Cursor erkennen, den Sie aber (wenn nötig) mit dem Befehl CURS OFF abschalten können.

WINDSAVE *(Speichert den Inhalt des aktuellen Fensters)*

WINDSAVE

Durch den Befehl WIND SAVE können Sie Ihre Fenster beliebig auf dem Bildschirm verschieben, ohne Ihr vorhandenes Display zu zerstören.

Wenn Sie diese Eigenschaft einmal aktiviert haben, dann werden alle Fenster, die Sie danach eröffnen, automatisch den gesamten Inhalt der darunterliegenden Fenster speichern. Wenn Sie ein Fenster dann schließen oder zu einer neuen Position bewegen, wird dieser Bereich neu gezeichnet.

Dabei ist wichtig, daß diese Option den Inhalt des aktuellen Fensters speichert und nicht desjenigen, das Sie mit WIND OPEN definieren.

Am Anfang Ihres Programms erscheint der Default-Bildschirm als aktuelles Fenster; mit 32 KB frißt er jedoch ganz massiv Speicher. Wenn Sie zum Beispiel den Hintergrund einer Dialogbox speichern wollen, dann würde ein Großteil dieses Speichers völlig verschwendet.

Die Lösung besteht darin, ein Musterfenster der erforderlichen Größe zu schaffen und es auf der Zone, die Sie speichern möchten, zu positionieren. Jetzt können Sie den Befehl WIND SAVE ausführen und wie üblich mit Ihrem Programm fortfahren.

Wenn Sie später Ihre Dialogbox aufrufen, dann sehen Sie, daß der darunterliegende Bereich als Teil Ihres Musterfensters gespeichert wurde. Nach dem Entfernen Ihrer Box wird es automatisch wieder erstellt.

BORDER *(Verändert die Umrandung des Fensters auf dem aktuellen Schirm)*

BORDER n,paper,pen

Mit dem Befehl BORDER können Sie die Umrandung des aktuellen Fensters definieren. Die Zahl n stellt dabei die Nummer der Stilart dar. Die Umrandung wird durch eine Reihe von Zeichen, die im Default-Font installiert sind, erstellt. Mit dem Zusatz für die Definition von Fonts (font definer accessory) können Sie auch Ihre eigenen Stilarten entwerfen.

Die Optionen paper und pen bieten Ihnen die freie Farbauswahl für Ihre Umrandung. Dabei liegen die zugelassenen Werte für die Umrandungsnummern zwischen 1 und 16.

Auch bei dieser Anweisung kann jeder der Parameter weggelassen werden, und die folgenden Befehle sind alle völlig erlaubt:

```
Border 2,,
Border 1,2,3
Border 2,,3 : Rem Nicht vergessen fehlende Werte durch Kommas zu ersetzen
```

TITLE TOP *(Definiert den oberen Titel für das aktuelle Fenster)*

TITLE TOP t$

Die Anweisung TITLE TOP plaziert die Zeichenkette t$ in die oberste Zeile des aktuellen Fensters. Dieser Titel wird jetzt am oberen Rand Ihres Fensters angezeigt. Auf diese Art können jedoch nur umrandete Fenster mit einem Titel versehen werden.

```
Windopen 5,1,1,20,10
Title Top "Fenster Nummer 5"
Wait Key
```

TITLE BOTTOM *(Definiert den unteren Titel für das aktuelle Fenster)*

TITLE BOTTOM b$

Dieser Befehl weist dem unteren Titel des aktuellen Fensters die Zeichenkette b$ zu.

```
Windopen 5,1,1,20,10
Title Bottom "Fenster Nummer 5"
Wait Key
```

WINDOW *(Ändert das aktuelle Fenster)*

WINDOW n

WINDOW aktiviert das Fenster mit der Nummer n als aktuelles Fenster. In diesem Fenster werden von nun an alle folgenden Text-Operationen durchgeführt. Wenn zuvor das automatische Speicher-System initialisiert wurde, wird dieses Fenster mit seinem gesamten Inhalt sofort neu gezeichnet. Laden Sie **BEISPIEL 8.6** aus dem MANUAL-Unterverzeichnis und sehen Sie sich das doch an einem Beispiel an.

=WINDON *(Gibt den Wert des aktuellen Fensters an)*

W=WINDON
WINDON gibt die Identifikationsnummer des derzeit aktiven Fensters an. Dazu folgendes Beispiel:

```
Windopen Rnd(12)+1,10,10,10,10
Print "Fenster Nummer";windon,"Aktiviert"
```

WIND CLOSE *(Schließt das aktuelle Fenster)*

WIND CLOSE

Mit dem Befehl WIND CLOSE löschen Sie das aktuelle Fenster. Wenn Sie zuvor den Befehl WIND SAVE eingegeben haben, dann wird das Fenster durch die gespeicherten Grafiken ersetzt, andernfalls wird der gesamte Bereich vollständig vom Bildschirm gelöscht.

```
Windopen 1,1,8,38,18,1 : Print "Drücke eine Taste und schließe dieses Fenster"
Wait Key
Wind Close
```

WINDMOVE *(Verschiebt ein Fenster)*

WIND MOVE X;Y

WINDMOVE verschiebt das aktuelle Fenster zu den Grafik-Koordinaten x,y. Wie bei der ursprünglichen Fenster-Definition wird die x-Koordinate zur nächstgelegenen 16-Pixel-Grenze auf- bzw. abgerundet. Zum Beispiel:

```
Wind Save
Wind Open 1,0,2,30,10,1
Wind Save
For M=1 to 100
   Pen Rnd(15) : Paper(15) : Print : Centre "Fenster verschieben!"
   Wind Move Rnd(30)+1,Rnd(100)+1
   Wait Vbl
Next M

Wind Save : Wind Open 1,0,2,10,5 : Wind Save
Curs Off : Paper 5 : Pen 0 : Clw
Print : Print "Verschiebe den" : Print "Maus-" : Print "Zeiger"
BEGIN:
```

```
On Error Goto STP
LOP:
WX=X Screen(X Mouse) : WY=Y Screen(Y Mouse)
If OX=WX and OY=WY Then Goto Else Wind Move WX,WY
OX=WX : OY=WY : Goto LOP
STP:
OX=WX : OY=WY: Resume BEGIN
```

Wenn Sie zuvor die Funktion für das Speichern des Fensters aktiviert haben, wird dieses Fenster am neuen Standort wieder angezeigt, andernfalls erscheint der Bildschirm völlig unverändert.

WINDSIZE *(Verändert die Größe des aktuellen Fensters)*

WIND SIZE sx,sy

Dieser Befehl verändert die Größe eines AMOS-Fensters. Die Einheiten zur Darstellung der neuen Abmessungen sx und sy beruhen auf der Größe eines Zeichens. Sx muß dabei durch zwei teilbar sein. Siehe auch **BEISPIEL 8.7.**

Wenn Sie zuvor den Befehl WIND SAVE eingegeben haben, dann wird der ursprüngliche Inhalt Ihres Fensters durch diese Anweisung neu gezeichnet. Wenn das neue Fenster kleiner als das vorherige ist, so gehen alle Teile des Bildes, die außerhalb der neuen Abmessungen liegen, verloren. Wenn Sie allerdings Ihr Fenster vergrößert haben, wird der Bereich, der den von Ihnen gespeicherten Teil umgibt, mit der aktuellen Hintergrundfarbe (paper colour) ausgefüllt. Beachten Sie dabei auch bitte, daß der Text-Cursor nach dem Befehl WIND SIZE immer auf die Koordinaten 0,0 zurückgesetzt wird.

CLW *(Löscht das aktuelle Fenster)*

CLW

CLW löscht den Inhalt des aktuellen Fensters und füllt es völlig mit der aktuellen Hintergrundfarbe aus. Hier ist ein Beispiel:

```
Rem Löscht Fenster Nummer W
Procedure CLEAR_WIN[W]
   WIND_OLD=Windon
   Window W : Clw
   Window WIND_OLD
End proc
```

Schieberegler

AMOS umfaßt drei Anweisungen, die es Ihnen ermöglichen, einen Standard-Schieberegler- Balken (Slider) auf dem Bildschirm darzustellen. Diese Slider können nicht direkt mit der Maus manipuliert werden. Wenn Sie einen funktionsfähigen Slider-

Balken erstellen möchten, müssen Sie eine kleine Basic-Routine schreiben, um diese Operation in Ihrem Hauptprogramm durchzuführen. AMOS ist jedoch so leistungsstark, daß Sie das im Handumdrehen durchziehen können. Wie Sie in **BEISPIEL 8.8** sehen können, ist das Ergebnis meist recht beeindruckend.

HSLIDER *(Zeichnet einen horizontalen Slider)*

HSLIDER x1,y1 TO x2,y2,Gesamt,Pos,Größe

Mit dieser Anweisung zeichnen Sie einen horizontalen Slider-Balken von x1,y1 bis x2,y2. x1,y1 sind dabei die Koordinaten der linken oberen Ecke und x2,y2 die Koordinaten des diagonal gegenüberliegenden Punktes.

Gesamt gibt die Anzahl der einzelnen Einheiten an, in die der Slider unterteilt wird. Jede Einheit stellt dabei einen Bestandteil des Objektes dar, das Sie durch den Slider steuern. So wird zum Beispiel im Editier-Fenster mit Gesamt die Anzahl der Zeilen des derzeitigen Programms angegeben. Die Größe jeder Einheit errechnet sich nach folgender Formel:

(X2-X1)/GESAMT

Pos ist die Position der Slider-Box vom Anfang des Sliders, dargestellt in den Einheiten, die Sie mit Gesamt festgelegt haben. Größe stellt die Länge der Slider- Box in den vorgenannten Einheiten dar. Dazu folgende Beispiele:

```
Hslider 10,10 to 100,20,100,20,5
Hslider 10,50 to 150,100,25,10,10
```

Laden Sie **BEISPIEL 8.9** und sehen Sie sich eine praktische Demonstration eines dieser Slider an.

VSLIDER *(Zeichnet einen vertikalen Slider)*

VSLIDER x1,y1 TO x2,y2,Gesamt,Pos,Größe

Die Anweisung VSLIDER ist mit HSLIDER so gut wie identisch. Mit diesem Befehl wird ein einfacher Slider von x1,y1 bis x2,y2 erstellt. x1,y1 und x2,y2 bestimmen dabei Position und Größe Ihres Sliders. Beispiele:

```
Vslider 10,10 To 20,100,100,20,5
Vslider 0,0 To 319,199,10,2,6
```

In **BEISPIEL 8.10** im MANUAL-Unterverzeichnis finden Sie ein Beispiel für einen funktionstüchtigen vertikalen Slider.

SET SLIDER *(Bestimmt das Füllmuster für einen Slider)*

SET SLIDER b1,b2,b3,pb,s1,s2,s3,sp

Dieser Befehl sieht auf den ersten Blick zwar sehr kompliziert aus, er ist aber eigentlich ganz einfach. SET SLIDER dient zur Eingabe der Farben und Füllmuster, die für die mit den Befehlen HSLIDER und VSLIDER erstellten Slider-Balken verwendet werden sollen.

b1,b2,b3 definieren die Farben für Zeichnung, Hintergrund und Rahmen (ink, paper, outline) des Hintergrunds der Box. pb wählt das Füllmuster, das in diesem Bereich eingesetzt werden soll.

Entsprechend werden mit s1,s2,s3 die Farben für die Slider-Box selbst bestimmt; durch sp wird das Füllmuster definiert. bp und sp können dabei ganz beliebige Füllmuster sein. Wie üblich bezieht sich ein negativer Wert auf ein Sprite-Bild aus der derzeit geladenen Sprite-Bank. Auf diese Art können Sie ganz außerordentlich farbenprächtige Slider-Boxen gestalten. Hier ist ein Beispiel dafür:

```
Centre "<Drücke eine Taste>" : Curs Off
Do
   B1=Rnd(15) : B2=Rnd(15) : B3=Rnd(15) : BP=Rnd(24)
   S1=Rnd(15) : S2=Rnd(15) : S3=Rnd(15) : SP=Rnd(24)
   Set Slider B1,B2,B3,BP,S1,S2,S3,SP
   Hslider 10,50 To 300,60,100,20,25
   Vslider 10,60 To 20,180,100,20,25
   Wait Key
Loop
```

oder Beispiel 8.13

Fonts

AMOS bietet Ihnen die Wahl zwischen zwei verschiedenen Fontarten; das sind Textfonts auf der einen und Grafikfonts auf der anderen Seite. Textfonts werden über die PRINT- und WINDOW-Befehle aktiviert. Man bezeichnet sie auch als Zeichensätze, und jedes AMOS Basic-Fenster kann über einen eigenen Zeichensatz verfügen. Die Grafikfonts sind flexibler, und es wird eine Vielfalt verschiedener Gestaltungsmöglichkeiten angeboten:

Grafik-Text

Ihr Amiga kann eine beeindruckende Vielzahl an verschiedenen Schriftarten darstellen. Die Original-Workbench-Diskette enthält acht attraktive Fonts in einer Reihe von Schriftgrößen, und wenn Sie Public-Domain-Software einsetzen, dann können Sie noch zahlreiche weitere Fonts wählen. Wenn Sie den Upgrade zu Workbench 1.3 mitgemacht haben, dann können Sie mit dem FED-Programm auf der Extras-Diskette auch Ihre eigenen Fonts gestalten.

AMOS-Basic unterstützt diese Fonts vollkommen. Der Text kann in jeder der verfügbaren Schriftarten an jeder beliebigen Stelle auf dem Bildschirm dargestellt werden.

Mit den AMOS-Fonts können Sie selbst die einfachsten Spiele noch aufpeppen, und

für das Erstellen der Lade-Schirme und Punktstandtabellen in Ihren Spielen sind sie einfach unbezahlbar. Deshalb können wir Ihnen nur raten, sie in Ihren AMOS-Basic-Programmen voll auszunutzen.

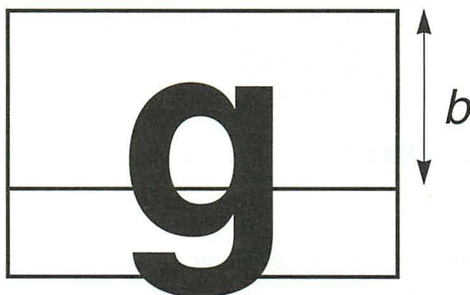
TEXT *(Druckt Grafik-Text)*

TEXT x,y,t$

Der Befehl TEXT positioniert die in der Zeichenkette t$ enthaltene Textzeile an den Grafik-Koordinaten x,y auf dem Bildschirm. Alle Koordinaten werden in Relation zur Grundlinie des Zeichens gemessen. Diese Grundlinie kann mittels der speziellen Funktion TEXT BASE bestimmt werden.

Normalerweise befindet sich die Grundlinie an der Unterkante des Zeichens, aber einige Kleinbuchstaben - wie g zum Beispiel - besitzen eine Unterlänge, die etwas über diesen Punkt hinausgeht. Das wird anhand des folgenden Diagramms verdeutlicht:

Die Grundlinie



Der Default für die Schriftarten ist auf Eight-point Topaz eingestellt, kann aber jederzeit über die Anweisung SET FONT geändert werden. Probieren Sie doch einmal das folgende Programm aus und sehen Sie sich selbst an, wie der Text an jedem beliebigen Pixel auf dem Bildschirm plaziert werden kann.

```
Do
   Ink Rnd(15)+1,Rnd(15): Text Rnd(320)+1,Rnd(198)+1,"AMOS Basic"
Loop
```

Bitte beachten Sie dabei auch, daß die Farbe Ihres Text mit dem Befehl INK und nicht - wie Sie vielleicht erwartet haben - mit PEN und PAPER bestimmt wird. Dies unterstreicht die Tatsache, daß der TEXT-Befehl im Grunde eine Grafik-Anweisung darstellt. Deshalb werden die durch Funktionen wie CUP$ hervorgerufenen Kontroll-Sequenzen auf dem

Bildschirm dargestellt, anstatt korrekt interpretiert zu werden.

Wenn der Text den Rand des aktuellen Fensters erreicht, findet kein automatischer Zeilenvorschub statt. Sollten Sie also versuchen, etwas zu auszugeben, das zu lang ist, so wird der Text sauber an der Bildschirmgrenze abgeschnitten. Das nachstehende Beispiel verdeutlicht dies.

```
Print String$("A",100): Text 0,100,String$("A",100)
```

GET FONTS *(Erstellt eine Liste aller verfügbaren Fonts)*

GET FONTS

Durch den Befehl GET FONTS wird eine interne Liste aller auf der gegenwärtig geladenen Start-Diskette verfügbaren Fonts erstellt. Diese Liste ist eine Voraussetzung für den Einsatz des SET FONTS Befehls, deshalb sollten Sie GET FONTS mindestens einmal aufrufen, bevor Sie versuchen, die gegenwärtig gewählten Fonts zu ändern. Mit der FONT$-Funktion können Sie dann den Inhalt dieser Liste überprüfen.

Achtung! Damit der Befehl GET FONTS funktioniert, muß Ihre aktuelle AMOS-Arbeitsdiskette immer eine Kopie des Standard-LIBS-Verzeichnisses samt Inhalt aufweisen. Sie sollten das nie vergessen, wenn Sie kompilierte oder "run-only"-Programme verteilen, weil sich AMOS-Basic sonst höchstwahrscheinlich umgehend verabschiedet, wenn Ihre Disketten die erforderlichen Dateien nicht aufweisen.

GET DISC FONTS *(Erstellt eine Liste der Fonts auf der Diskette)*

GET DISC FONTS

Dieser Befehl stimmt mit der obengenannten Anweisung GET FONTS überein, allerdings mit dem Unterschied, daß hier die Fonts auf der Diskette gesucht werden. Diese Fonts befinden sich im FONTS-Verzeichnis auf Ihrer aktuellen Boot- Diskette. Wenn Sie Ihre eigenen Fonts mit AMOS-Basic einsetzen möchten, dann müssen Sie diese auf Ihre normale Start-Diskette kopieren. In Ihrem Amiga- Handbuch ist dieser Ablauf genau beschrieben.

GET ROM FONTS *(Erstellt eine Liste der ROM-Fonts)*

GET ROM FONTS

Der Befehl GET ROM FONTS stellt eine Liste der Fonts zusammen, die in die Amiga ROM-Chips eingebaut sind. Gegenwärtig gibt es davon nur zwei: 8 Pixel (Höhe) Topaz und 9 Pixel Topaz.

```
Get Rom Fonts
```

```
Set Font 1
Text 100,100,"Topaz 9"
Set Font 2
Text 100,120,"Topaz 8"
```

=FONT$ *(Gibt Einzelheiten zu den verfügbaren Fonts an)*

a$=FONT$(n)

Durch FONT$ erhalten Sie ein Kette aus 38 Zeichen, die den Font mit der Nummer n beschreibt. Mit dieser Funktion können Sie die Font-Liste, die Sie mit einem der GET FONT-Befehlen erstellt haben, genauer untersuchen. n stellt dabei die Nummer des Fonts dar, den Sie sich ansehen möchten.

a$ enthält eine Anzahl von Zeichen, die Namen und Art Ihres Fonts angeben. Wenn ein Font nicht existiert, wird für a$ der Null-Wert "" angegeben. Normalerweise wird eine Zeichenkette in folgendem Format dargestellt:

Zeichen	Beschreibung
1-29	Font-Name
30-33	Font-Höhe
34-37	Identifizierung (entweder auf "Disc" oder "ROM" gestellt)

Die Datei **BEISPIEL 8.11** im MANUAL-Unterverzeichnis auf Ihrer AMOS-Programmdiskette enthält hierzu ein Beispiel. Sie finden darin eine Reihe nützlicher Prozeduren, die Sie beliebig in Ihren Programmen einsetzen können.

SET FONT *(Wählt einen Font für den Einsatz mit der TEXT-Anweisung)*

SET FONT n

SET FONT ändert den Zeichensatz, auf den der TEXT-Befehl zugreift, auf den Font mit der Nummer n. Wenn der Font auf der Diskette gespeichert ist, wird er damit automatisch in den Speicher Ihres Amiga geladen. Gleichzeitig werden alle zuvor gewählten Fonts, die jetzt nicht mehr eingesetzt werden, entfernt.

Anhand des folgenden einfachen Beispiels können Sie sehen, wie dieser Befehl funktioniert:

```
Get Fonts
Set Font 2 : Text 100,100,"AMOS" : Set Font 1 : Text 100,120,"Basic"
```

Sie sollten hier beachten, daß die Anweisung GET FONTS vor der Auswahl des Fonts aufgerufen wird. Der Grund dafür ist, daß SET FONT auf die mit GET FONTS erstellte Font-Liste zugreift. In **BEISPIEL 8.12** finden Sie eine umfassende Demonstration dieses Befehls.

SET TEXT *(Wahl des Text-Attributs)*

SET TEXT Attribut

Mit dem Befehl SET TEXT können Sie die Darstellung eines Fonts verändern. Sie können dabei unter den folgenden Attributen wählen: unterstrichen, fett und kursiv. Attribut ist ein Bitmuster mit folgendem Format:

Bit	Effekt
0	Wählen Sie dieses Bit, um Ihre Zeichen zu unterstreichen.
1	Stellt die Zeichen im Fettdruck dar.
2	Jetzt schreiben Sie *kursiv*.

Durch Einstellen der entsprechenden Bits in diesem Muster können Sie zwischen acht verschiedenen Schriftarten wählen. Hier ist ein Beispiel, das Sie nun in Ihren Computer eingeben können:

```
Colour 2,0 : Colour 1,$fff : Flash Off : Get Rom Fonts : Set Font 1
For S=0 To 7 : Set Text S : Text 100,S*20+20,"AMOS Basic" : Next S
```

=TEXT STYLES *(Gibt das aktuelle Text-Attribut an)*

s=TEXT STYLES

Diese Funktion gibt das Text-Attribut an, das Sie zuvor mit dem Befehl SET TEXT bestimmt haben. Das in s dargestellt Ergebnis ist eine Bitmap im selben Format wie SET TEXT.

=TEXT LENGTH *(Gibt die Länge eines Grafiktext-Abschnitts an)*

w=TEXT LENGTH(t$)

Die Funktion TEXT LENGTH gibt die Breite der Zeichenkette a$ im aktuellen Font in Pixel an. Die Breite eines Zeichens hängt dabei von der Schriftgröße Ihres Fonts ab. Dazu kommt, daß Proportionalschriften wie Helvetica den einzelnen Zeichen eine unterschiedliche Breite zuweisen. Hier ist ein einfaches Beispiel:

```
T$="Zentrierter Text"
L=Text Length(T$)
Text 160-L/2,100,T$
```

=TEXT BASE *(Gibt die aktuelle Grundlinie an)*

b=TEXT BASE

Mit dieser Funktion können Sie die Position der Grundlinie Ihres Fonts abfragen. Die Grundlinie ist die Anzahl der Pixel zwischen der Oberkante eines Zeichens und dem Punkt, an dem es auf dem Bildschirm erscheint. Im Grunde ist das dem "Hot Spot" eines Sprites oder BOBs ähnlich.

Installieren von neuen Fonts

Wenn Sie in AMOS-Basic Ihre eigenen Fonts anwenden möchten, dann müssen Sie sie auf einer Kopie Ihrer AMOS-Programmdiskette speichern. Dabei gehen Sie grundsätzlich so vor:

- Kopieren Sie die erforderlichen Font-Dateien in das FONTS:-Verzeichnis Ihrer Boot-Diskette.

- Im Extra-Handbuch, das mit dem Workbench 1.3 Upgrade geliefert wird, können Sie noch weitere Informationen hierzu finden.

Fehlersuche

Es ist zwar nicht garnicht so schwer, die Fonts anzuwenden, aber wenn man nicht vorsichtig ist, kann man doch in manche Falle tappen. Wir wollen Ihnen hier eine Reihe von Lösungen zu den häufigsten Problemen nennen.

Problem: GET FONTS scheint die Fonts auf der aktuellen Diskette nicht anzusprechen.
Lösung: Wahrscheinlich haben Sie die ursprüngliche Boot-Diskette aus dem Default-Laufwerk genommen. Die Library-Routinen des Amiga suchen nach dem FONTS:-Verzeichnis auf Ihrer Start-Diskette. Dieses Vorgehen können Sie mit dem ASSIGN-Programm im UTILITIES-Verzeichnis ändern. Weitere Einzelheiten dazu können Sie auch unter dem Befehl GET DISC FONTS finden.
Problem: GET FONTS bringt Ihren Amiga total zum Absturz.
Lösung: Dieses Problem kann leicht entstehen, wenn Sie Programme im kompilierten oder "run-only"-Format erstellen. Damit der Befehl GET FONTS funktioniert, muß die DISCFONT-Library im LIBS-Verzeichnis sein. Wenn Sie vergessen, dieses Unterverzeichnis auf die Disketten zu kopieren, die Sie ausgeben, kommt es zu einem System-Fehler, der den Amiga abstürzen lassen kann.
Problem: Auf den SET FONT Befehl erscheint die Fehlermeldung fonts not examined.
Lösung: Rufen Sie am Anfang Ihres Programm zusätzlich noch GET FONTS auf. Dadurch wird eine Liste aller gegenwärtig verfügbaren Fonts erstellt, auf die der Befehl SET FONTS zugreifen kann.



9: Mathematische Funktionen

AMOS-Basic enthält eine Vielzahl der gebräuchlichsten mathematischen Funktionen. Um Speicher zu sparen, verwendet AMOS die normalen Library- Routinen des Amiga. Die entsprechenden Libraries werden automatisch von Ihrer Workbench-Diskette geladen, wenn Sie eine der folgenden Funktionen in einer Arbeitssitzung das erste Mal aufrufen. Sie sollten deshalb sicherstellen, daß die Diskette, die sich im Laufwerk befindet, die Datei MATHTRANS.LIBRARY im LIBS- Unterverzeichnis aufweist.

Trigonometrische Funktionen

Mit den trigonometrischen Funktionen erhalten Sie eine nützliche Reihe mathematischer Werkzeuge. Sie können vielseitig eingesetzt werden, angefangen bei Unterricht und Studium bis zur Erstellung komplexer musikalischer Wellenformen.

DEGREE *(Einsatz von Grad)*

DEGREE

Normalerweise werden alle Winkel im Bogenmaß (Radian) angegeben. Da es jedoch ziemlich schwer ist, mit Winkeln im Bogenmaß zu arbeiten, können Sie AMOS die Anweisung geben, alle Winkel in Grad zu akzeptieren. Wenn Sie diese Funktion einmal aktiviert haben, gilt sie für alle folgenden Einsätze der trigonometrischen Funktionen.

```
Degree
Print Sin(45)
```

RADIAN *(Verwendet das Bogenmaß)*

RADIAN

Der Befehl RADIAN teilt AMOS mit, daß alle von nun eingegebenen Winkel in Bogenmaß angegeben werden - und das ist auch der Default.

=PI# *(Die Konstante O)*

a#=PI#

Durch diese Funktion erhält man die als PI bekannte Zahl, die das Ergebnis der Division eines Kreisdurchmessers durch seinen Umfang darstellt. PI wird bei den meisten

trigonometrischen Funktionen zur Berechnung eines Winkels eingesetzt. Beachten Sie bitte, daß das Zeichen # ein Bestandteil des Token-Namens ist. Damit soll eine Verwechslung mit den Namen Ihrer eigenen Variablen vermieden werden.

=SIN *(Sinus)*

s#=SIN(a)
s#=SIN(a#)

Die SIN-Funktion berechnet den Sinus des in a angegebenen Winkels. Beachten Sie, daß Sie durch diese Funktion immer eine Fließkommazahl erhalten. Zum Beispiel:

```
Degree
For X=0 To 319
   Y#=Sin(x)
   Plot X,Y#*50+100
Next X
```

Siehe dazu auch den Befehl HSIN.

=COS *(Cosinus)*

c#=COS(a)
c#=COS(a#)

Die Cosinus-Funktion berechnet den Cosinus eines Winkels. Normalerweise werden alle Winkel im Bogenmaß (Radian) gemessen. Das können Sie durch den Befehl DEGREE verändern. Fügen Sie jetzt die folgenden beiden Zeilen zu dem oben angeführten Beispiel hinzu. Dabei müssen Sie sie zwischen den Anweisungen Plot und Next einsetzen.

```
Y#=Cos(X)
Plot X,Y#*50+100
```

Siehe auch die Befehle ACOS, HCOS.

=TAN *(Tangens)*

t#=TAN(a)
t#=TAN(a#)

TAN generiert den Tangens eines Winkels. Hier sind einige Beispiele:

```
Degree : Print Tan(45)
0.9999998
Radian : Print Tan(PI#/8)
0.4141
```

Siehe auch ATAN, HTAN.

=ACOS *(Arcuscosinus)*

c#=ACOS(n#)

Die ACOS-Funktion nimmt eine Zahl zwischen -1 und +1 und berechnet den Winkel, der erforderlich wäre, um diesen Wert mit COS zu generieren.

Also, wenn X#=COS(Winkel) dann WINKEL=ACOS(X#).

Beachten Sie hier bitte, daß wir die Funktion ASIN nicht miteinbezogen haben, da man Sie nicht wirklich braucht. Man kann die Berechnungen nämlich ganz leicht mit folgender Formel durchführen:

ASIN(X)=90-ACOS(X) : Rem In Grad gemessen
ASIN(X)=1.5708-ACOS(X) : Rem Im Bogenmaß

Beispiel:

```
A#=Cos(45)
Print Acos(A#)
```

Siehe auch die Funktionen COS, HCOS.

=ATAN *(Arcustangens)*

t#=ATAN(n#)

ATAN weist den Arcustangens einer Zahl aus. Hier ist ein Beispiel:

```
Degree : Print Tan(2)
0.03492082

Degree : Print Atan(0.03492082)
2
```

=HSIN *(Sinus Hyperbolicus)*

s#=HSIN(a)
s#=HSIN(a#)

HSIN berechnet den Sinus Hyperbolicus des Winkels a.

=HCOS *(Cosinus Hyperbolicus)*

c#=HCOS(a)
c#=HCOS(a#)

HTAN gibt den Cosinus Hyperbolicus des Winkels a an.

=HTAN *(Tangens Hyperbolicus)*

t#=HTAN(a)
t#=HTAN(a#)

HTAN gibt den Tangens Hyperbolicus des Winkels a an.

Mathematische Standard-Funktionen

=LOG *(Logarithmus)*

r#=LOG(v)
r#=LOG(v#)

LOG gibt den Logarithmus des Ausdrucks in v# auf der Basis 10 (log10) an. Hier sind einige Beispiele:

```
Print Log(10)
V#=Log(100)
```

=EXP *(Exponentialfunktion)*

r#=EXP(e#)

Berechnet den Exponentialwert von e#. Dazu folgendes Beispiel:

```
Print Exp(1)
2.71828
```

=LN *(Natürlicher Logarithmus)*

r#=LN(l#)

LN berechnet den natürlichen Logarithmus von l#. Hier einige Beispiele:

Print Ln(10)
2.30258
R#=Ln(100) : Print R#
4.60517

=SQR *(Quadratwurzel)*

s#=SQR(v)
s#=SQR(v#)

SQR berechnet die Quadratwurzel einer Zahl. Beispiele:

Print Sqr(9)
3
Print Sqr(11.1111)
3.33333

=ABS *(Absoluter Wert)*

r=ABS(v)
r#=ABS(v#)

ABS weist den absoluten Wert von v aus, ohne dabei das Vorzeichen in Betracht zu ziehen. Hier ein Beispiel:

Print Abs(-1),Abs(1)
1 1

=INT *(Rechnet eine Fließkommazahl in eine ganze Zahl um)*

i=INT(v#)

INT rundet die Fließkommazahl in v auf die nächstliegende ganze Zahl ab. Beispiele:

Print Int(1.25)
1
Print Int(-1.25)
-2

=SGN *(Ermittelt das Vorzeichen einer Zahl)*

s=SGN(v)
s=SGN(v#)

Durch SGN erhält man einen Wert, der das Vorzeichen einer Zahl darstellt. Es gibt dafür drei Möglichkeiten:

-1 wenn v negativ ist
0 wenn v gleich Null ist
1 wenn v positiv ist

Das Hervorrufen von Zufalls-Sequenzen

=RND *(Generiert Zufallszahlen)*

v=RND(n)

RND generiert eine ganzzahlige Zufallszahl, die zwischen O und n einschließlich liegt. Wenn n aber kleiner als Null ist, weist RND dafür den letzten hervorgerufenen Wert aus. Das kann bei der Suche nach Bugs sehr hilfreich sein. Beispiele:

```
Do
   C=Rnd(15) : X=Rnd(320) : Y=Rnd(200) : Ink c : Text X,Y "ZUFALL"
Loop
```

RANDOMIZE *(Festlegen der Basis der Zufallszahlen)*

RANDOMIZE seed (Basis)

In der Praxis stellen die durch die RND-Funktion hervorgerufenen Zahlen jedoch keine echten Zufallszahlen dar. Sie werden intern anhand einer komplexen mathematischen Formel berechnet. Der Anfangspunkt für diese Berechnung beruht auf einer Zahl, die man als seed (Basis) bezeichnet. Jedesmal, wenn Sie AMOS- Basic in Ihren Computer laden, wird für diese Zahl ein Standardwert festgelegt. Deshalb ist die Sequenz von Zahlen, die RND generiert, jedesmal gleich, wenn Sie Ihre Spiele ablaufen lassen.

Das mag ja für Spielhallen-Spiele ganz in Ordnung sein, wenn Sie aber ein Kartenspiel wie Poker simulieren möchten, gäbe es ernsthafte Schwierigkeiten. Dieses Problem können Sie mit dem Befehl RANDOMIZE lösen, denn nun können Sie die Basis direkt festlegen.

Sie können hier jede beliebige Zahl eingeben. Jede Basis generiert ihre ganze eigene Sequenz von Zahlen.

Um echte Zufallszahlen hervorzurufen, müssen Sie die Basis bei jedem Spiel ändern können. Das ist durch die TIMER-Anweisung möglich:

Randomize Timer

TIMER ist eine Basic-Funktion, die die Zeitspanne angibt, die seit dem Einschalten des Amiga für die gegenwärtige Arbeitssitzung verstrichen ist. Diese Zeitspannen werden in Einheiten von Fünfzigstel einer Sekunde gemessen.

Am besten setzen Sie diese Anweisung sofort nach dem der Anwender Informationen in den Computer eingegeben hat ein. Dabei reichen ganz einfache Dinge aus, wie zum Beispiel die Notwendigkeit, eine Taste zu drücken, bevor das Spiel beginnt, da es natürlich unmöglich ist, auf die nächste Fünfzigstelsekunde genau vorauszusagen, wann der Anwender eine bestimmte Taste drückt.

Vorausgesetzt, daß der Inhalt von TIMER einigermaßen gut verändert werden kann, dann ist die Sequenz der durch die RND-Funktion generierten Zahlen in jedem Spiel anders.

Das Manipulieren von Zahlen

=MAX *(Weist den Größten von zwei Werten aus)*

r=MAX(x,y)
r#=MAX(x#,y#)
r$=MAX(x$,y$)

MAX vergleicht zwei Ausdrücke miteinander und gibt dann den Größten an. Diese Ausdrücke können sich aus Zahlen oder Zeichenketten zusammensetzen. Dabei dürfen Sie die verschieden Arten jedoch nicht in einer Anweisung kombinieren.

Print Max(10,4)
10
Print Max("Hallo","Tag")
Tag

=MIN *(Gibt den kleineren von zwei Werten an)*

r=MIN(x,y)
r#=MIN(x#,y#)
r$=MIN(x$,y$)

Die MIN-Funktion gibt den kleineren Wert von zwei Ausdrücken an. Die Ausdrücke können dabei aus Zeichenketten, ganzen oder reellen Zahlen bestehen. Sie können jedoch nur gleichartige Werte miteinander vergleichen. Hier sind einige Beispiele:

A=Min(10,4) : Print A
4
Print Min("Hallo","Tag")
Hallo

SWAP *(Vertauscht den Inhalt zweier Variablen)*

SWAP x,y
SWAP x#,y#
SWAP x$,y$

Vertauscht die Daten von zwei Variablen gleichen Typs. Das entspricht folgender Anweisung:

DUMMY=X : X=Y : Y=DUMMY

Beispiel:

A=10 : B=40 : Swap A,B : Print A,B

FIX *(Legt die Genauigkeit derFließkommazahlen fest)*

FIX(n)

FIX verändert die Art, wie Ihre Fließkommazahlen auf dem Bildschirm oder in einem Ausdruck dargestellt werden. Es gibt hier vier Möglichkeiten.
Für 0<n<16 gibt n die Anzahl der Zahlen an, die nach dem Komma (Dezimalpunkt) erscheinen sollen.

Für n>16 wird der Ausdruck proportional dargestellt und alle nachfolgenden Nullen erscheinen nicht.

Für n=16 wird wieder das normale Format eingestellt.

Für n<0 werden alle Fließkommazahlen im Exponentialformat dargestellt, und der absolute Wert von n(ABS(n)) bestimmt die Anzahl der Stellen nach dem Komma (Dezimalpunkt). Dazu folgende Beispiele:

Fix (2) : Print PI# : Rem Begrenzt die Anzahl auf zwei Stellen nach dem Komma.
Fix (-4) : Print PI# : Rem Bestimmt Exponential-Modus mit vier Stellen nach dem Komma.
Fix (8) : Print PI# : Rem Zurück zum normalen Modus.

Achtung: Diese Funktion unterscheidet sich vollkommen vom gleichlautenden Befehl in Amiga Basic.

DEF FN *(Schafft eine vom Anwender definierte Funktion)*

DEF FN Name [(Liste)]=Ausdruck

Mit dem Befehl DEF FN können Sie in einem AMOS Basic-Programm Ihre eigenen Anwender-Funktionen definieren. So können dann häufig benötigte Werte schnell und leicht berechnet werden.
Name ist der Name der Funktion, die Sie definieren möchten. Liste stellt eine Reihe von Variablen dar, die durch Kommas voneinander abgetrennt werden. Nur die Art der Variablen ist hier von Bedeutung. Wenn Sie Ihre Funktion aufrufen, werden die Variablen, über die Sie Zugang erhalten, automatisch an den entsprechenden Positionen ersetzt.

Der Ausdruck kann jede der AMOS Standardfunktionen enthalten. Auch hier steht Ihnen, wie für alle Basic-Ausdrücke, nur eine Zeile in Ihrem Programm für einen Ausdruck zur Verfügung.

Die neue Funktion kann mit der FN-Anweisung aufgerufen werden.

FN *(Ruft eine vom Anwender definierte Funktion auf)*

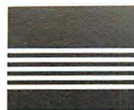
FN Name [(Variablenliste)]

FN führt eine mit DEF FN definierte Funktion aus. Hier sind einige einfache Beispiele dafür:

```
Def Fn Asin(X)=90-Acos(X)
Degree
Print Fn Asin(0.5)
30

Def Fn SLICE (A$,X,Y)=Mid$(A$,X,Y)
Print Fn SLICE("Hallo",2,3)
ell
```

Sie sehen hier, wie wir die Funktion mit DEF FN definiert haben, bevor wir sie in unserem Programm eingesetzt haben. Das ist nämlich unbedingt erforderlich.



10: Bildschirme

Ihr Commodore Amiga kann wirklich atemberaubende Bilder darstellen. Mit AMOS-Basic können Sie diese Bilder nun durch eine beeindruckende Vielzahl von Befehlen zur Bildschirm-Animation direkt in Ihre Spiele einbauen.

Jeder Grafik-Modus wird genau gleich behandelt. Deshalb ist es genauso leicht, einen HAM-Bildschirm mit 4096 Farben wie eine normale 16-Farbanzeige hervorzurufen. Sie können auch über das Dual-Playfield-System zwei getrennte Bildschirme miteinander verbinden. Damit können Sie so erstaunliche Parallax- Effekte wie in den Verkaufsschlagern Xenon II und Silkworm erstellen.

Der Default-Bildschirm

Wenn Sie ein AMOS-Basic-Programm ablaufen lassen, wird der Default-Bildschirm stets als Bildschirm 0 geöffnet. Er stellt die Standardanzeige dar, auf der alle Ihre normalen Zeichenschritte durchgeführt werden.

Der System-Default ist ein Bildschirm mit 16 Farben, der über die Abmessungen 320x200 verfügt. Diese Angaben können Sie leicht aus Ihrem Programm heraus verändern. Darüber hinaus können Sie mit dem starken Befehl SCREEN OPEN noch bis zu sieben weitere Bildschirme definieren.

DISPLAY HEIGHT *(Zeigt die maximale Bildschirmhöhe an)*

=DISPLAY HEIGHT

Dieser Befehl weist für PAL 311 und für NTSC 263 aus.

NTSC *(Zeigt die Art der aktuellen Anzeige an)*

=NTSC

Diese Anweisung zeigt TRUE (= Richtig) an, wenn das System im NTSC-Modus ist, und FALSE (= Falsch), wenn PAL eingestellt ist. Somit ist dieser Befehl für die Entwicklung von internationaler Software ideal!

Musik läuft bei PAL und NTSC mit genau derselben Geschwindigkeit ab. Da die Aktualisierungsrate des Bildschirms mit den Musikroutinen verbunden ist, muß die Musik verlangsamt werden, wenn sie auf einem NTSC-System abgespielt werden soll. Bei NTSC wird der Bildschirm nämlich 60mal pro Sekunde aktualisiert, während PAL-Bildschirme nur 50mal pro Sekunde aktualisiert werden.

AMAL wird ebenfalls durch die Interrupt-Routine gesteuert, wird jedoch zur Anpassung an die PAL-Geschwindigkeit nicht verlangsamt. Sie sollten sich daher nicht darauf verlassen, daß Musik und Animation aufgrund der Geschwindigkeit, mit der sie ablaufen, synchron sind. Überprüfen Sie, daß eine bestimmte Animationssequenz erreicht ist, oder daß eine bestimmte Note der Musik gespielt wurde. Mit dieser Technik können Sie sicherstellen, daß die Software auf allen Systemen richtig abläuft.

Seien Sie auch vorsichtig, wenn das Timing bei Ihren Programmen sehr eng ist. Um eine Animationsgeschwindigkeit von 50 Bildern pro Sekunde (60 ist etwas schneller) zu erreichen, kann das Programm auf einem amerikanischen Fernsehschirm flimmern!

Das Definieren eines Bildschirms

SCREEN OPEN *(Öffnen eines Bildschirms)*

SCREEN OPEN n,w,h, nc, Modus

SCREEN OPEN öffnet einen Bildschirm und reserviert Speicherplatz für ihn. Dieser neue Bildschirm dient jetzt als Ziel für alle folgenden Texteingaben und Grafik- Operationen in Ihrem Programm.

n ist die Identifikationsnummer des Bildschirms, den Sie mit dieser Anweisung eröffnen. Sie können hier Werte von 0 bis 7 eingeben. Wenn ein Schirm mit dieser Nummer bereits existiert, dann wird er von Ihrer neuen Definition völlig ersetzt.

w gibt die Breite des Bildschirms in Pixel an. Dabei müssen Sie sich allerdings nicht auf die tatsächliche Größe Ihres Anzeigebereichs beschränken. Es ist vollkommen in Ordnung, übergroße Bildschirme zu definieren, die dann mit dem Befehl SCREEN OFFSET manipuliert werden können.

Auf dieselbe Art bestimmen Sie mit *h* die Höhe Ihres Bildschirms. Vorausgesetzt, Sie haben genug Speicher zur Verfügung, können Sie leicht Bildschirme erstellen, die wesentlich größer sind als der sichtbare Anzeigebereich. Sie können trotzdem alle normalen Bildschirm-Operation problemlos durchführen. So können Sie zum Beispiel Ihre Bilder außerhalb des Anzeigebereichs konstruieren, und sie dann mit dem Befehl SCREEN OFFSET ins Blickfeld scrollen.

nc fragt die Anzahl der Farben ab, die für den neuen Bildschirm benötigt werden. Der Bereich der verfügbaren Farben bewegt sich zwischen 2 und 64 (Extra-Half- Bright). Sie haben allerdings auch Zugang zum Amiga HAM-Modus (Hold and Modify) mit einem Wertebereich von 4096 Farben.

Mit *Modus* können Sie die Breite der einzelnen Punkte auf dem Bildschirm festlegen. Der Amiga unterstützt entweder eine Bildschirmbreite von 320 oder 640 Pixel. Die gewünschte Breite stellen Sie ein, indem Sie für mode entweder LOWRES (0) d.h. Low Resolution (Niedrige Auflösung) oder HIRES ($8000) d.h. High Resolution (Hohe Auflösung) wählen.

Hier ist eine Reihe möglicher Bildschirm-Optionen mit Angaben dazu, wieviel Speicher sie jeweils benötigen.

Farben	Auflösung	Speicher	Bemerkungen
2	320x200	8 KByte	PAPER=0 PEN=1 Cursor=1, f1lash: keine Angabe
	640x200	16 KByte	PAPER=0 PEN=1 Cursor=1, flash: keine Angab

Farben	Auflösung	Speicher	Bemerkungen
4	320x200	16 KByte	PAPER=1 PEN=2 Cursor=3, flash=3
	640x200	32 KByte	: : :
8	320x200	24 KByte	PAPER=1 PEN=2 Cursor=3, flash=3
	640x200	48 KByte	
16	320x200	32 KByte	Das sind die Angaben für den Bildschirm 0(Default)
	640x200	64 KByte	
32	320x200	40 KByte	
64	320x200	48 KByte	Extra-Half-Bright-Modus
4096	320x200	48 KByte	HAM- (Hold and Modify) Modus

Beachten Sie bitte, daß sich die Angaben zur Speichergröße nur auf Standardbildschirme beziehen. Wenn Sie größere oder breitere Schirme eröffnen, ist der benötigte Speicherplatz natürlich erheblich größer. Der Bildschirm 0 (Default) entspricht den Angaben:

```
Screen Open 0,320,200,16,Lowres
```

Und hier folgen noch weitere Beispiele für AMOS-Bildschirme:

```
Rem Eröffnet einen 640x200 Hires-Schirm mit 8 Farben
Screen open 1,640,200,8, Hires

Rem Eröffnet Schirm 2 als HAM-Schirm
Screen open 2,320,256,4096,Lowers

Rem Eröffnet Schirm 3 als großen Schirm mit 8 Farben
Rem Es sind jeweils nur die ersten 320x256 sichtbar.
Screen open 3,500,400,8,Lowres

Rem Diese Demo eröffnet 8 Schirme und druckt auch auf sie alle aus!
Curs Off : Cls 13 : Paper 13
Print : Centre "Ich bin Schirm 0 hinten!"
For A=1 To 7
   Screen open A,320,20,16,Lowres
   Curs Off
   Cls A+5
   Paper A+5
   Centre "Ich bin Schirm"+Str$(A)
   Screen Display A,,50+A*25,,8
Next A
```

Siehe hierzu auch die Befehle SCREEN OFFSET, SCREEN DISPLAY und VIEW.

SCREEN CLOSE *(Schließt einen Bildschirm)*

SCREEN CLOSE n

Mit dem Befehl SCREEN CLOSE können Sie den Bildschirm mit der Nummer *n* schließen und den von ihm benutzten Speicherbereich für den Rest Ihres Programms verwenden.

AUTO VIEW ON/OFF *(Steuert den Sicht-Modus)*

AUTO VIEW OFF

Wenn Sie mit SCREEN OPEN einen Bildschirm eröffnen, wird der neue Schirm normalerweise sofort angezeigt. In der Initialisierungsphase Ihres Programmes ist das aber manchmal nicht wünschenswert.

Mit dem Befehl AUTO VIEW OFF können Sie den Aktualisierungsprozeß nun vollkommen steuern. Das automatische Anzeigesystem wird völlig ausgeschaltet, und Sie können die Bildschirmanzeige mit der VIEW-Anweisung an der passenden Stelle in Ihrem Programm aktualisieren.

AUTO VIEW ON

Aktiviert die automatische Aktualisierung der Bildschirmanzeige. In diesem Modus wird jede Änderung Ihres Schirm sofort auf Ihrem Fernsehschirm angezeigt.

DEFAULT *(Setzt den Bildschirm wieder auf den Default zurück)*

DEFAULT

Alle offenen Bildschirme werden geschlossen und die Anzeige wird wieder auf die ursprünglichen Default-Parameter zurückgesetzt. Dazu ein Beispiel:

```
Load Iff"AMOS_DATA:IFF/AMOSPIC.IFF",0
Wait Key
Default
```

VIEW *(Zeigt die aktuellen Bildschirm-Parameter an)*

VIEW

VIEW zeigt alle Veränderungen der aktuellen Bildschirm-Parameter bei dem nächsten Vertikal Blank an. Sie müssen diesen Befehl jedoch nur eingeben, wenn AUTOVIEW auf OFF gestellt ist.

Besondere Bildschirm-Modi

Die Farbe jedes Punktes auf dem Bildschirm wird von einem Wert bestimmt, der in einem der 32 Amiga-Farbregister gespeichert ist. Jedes Register kann aus einer Auswahl aus 4096 verschiedenen Farben geladen werden.

Sie denken nun vielleicht, daß 32 Farben eine ziemlich große Anzahl ist, vor allem, wenn man an den ST-Standard denkt, aber den Entwicklern des Amiga war das noch lange nicht genug. Die einfachste Lösung wäre es wahrscheinlich gewesen, einfach die Anzahl der Farbregister zu erhöhen, aber das wurde schnell aus Kostengründen abgelehnt.

Stattdessen wurden zwei spezielle Grafik-Modi erfunden, die die vorhandenen Register ganz clever ausnutzen um die maximale Anzahl an Farben auf den Bildschirm zu bringen.

Sie sind wahrscheinlich schon auf diese Modi gestoßen, es handelt sich hier um die berüchtigten Extra-Half-Bright- und HAM-Modi. AMOS-Basic unterstützt beide vollkommen. Das wollen wir nun kurz erklären.

Der Extra-Half-Bright-Modus (EHB)

Der Extra-Half-Bright-Modus verdoppelt die maximale Anzahl der Farben auf dem Bildschirm zu einer Gesamtzahl von 64. Das funktioniert durch die Generierung von zwei Farben für jedes der 32 möglichen Farbregister.

Die ersten 32 Farben laden den Farbwert direkt aus einem der Register. Jedes Register enthält einen Wert zwischen 0 und 4095, der die genaue Schattierung der Endfarbe bestimmt.

Die zweite Farbgruppe, mit den Nummern 32 bis 63, greift auf eines der vorherigen Register zu und teilt den enthaltenen Wert durch zwei. So entstehen 32 zusätzliche Farben, die genau halb so leuchtend sind wie die normalen Farbregister.

Angenommen, die Farbe Nummer Null enthält einen Wert von $FFF (Weiß). Für die Farbe Nummer 32 würde nun ein Wert von $777 (Grau) angezeigt. Die Schattierung der Farben von 32 bis 63 kann anhand der folgenden einfachen Formel ermittelt werden:

Farbe n=(Farbe n-32)/2

Wir sehen uns dieses Prinzip nun an folgendem Beispiel an:

```
Screen Close 0
Screen Open 2,320,167,64,Lowres : Flash Off
For I=1 To 32
   Ink I
   Bar 0,(I-1)*5 To 160,(2+I-1)*5
   Ink I+32
   Bar 160,(I-1)*5 To 319,(2+I-1)*5
Next I
```

Um den EHB-Modus vollständig auszuschöpfen, müssen Sie die leuchtendsten Farben Ihrer Palette in die 32 Register laden. Dann erhalten Sie automatisch ein Reihe von abgeschwächten Schattierungen für die Farben 32-63.

Abgesehen von der Farbpalette sind EHB-Bildschirme identisch zu allen anderen Bildschirm-Modi. Ihr Einsatz ist praktisch unbeschränkt. Sie können in diesem Modus sogar BOBs erstellen!

Bildschirm-Interlace

Einen Interlace-Bildschirm können Sie mit der folgenden Syntax öffnen:

SCREEN OPEN 0,320,200,16,Laced [+Hires] [+Lowres]

LACED ist dabie eine Funktion, die 4 ausweist.

Achtung: Sobald ein Schirm im Interlace-Modus geöffnet wird, übernehmen alle Bildschirme diesen Modus. Jedoch nur bei dem Bildschirm, der mit LACED geöffnet wird, findet echtes Interlacing statt. Bei allen anderen Schirmen werden nur die Linien auf dem Bildschirm verdoppelt.

Der Interlace-Modus ist ideal zur Darstellung von Bildern, die Animation läuft jedoch nur mit halber Geschwindigkeit. Man sollte im Interlace-Modus keine Spiele schreiben!

Nachdem der letzte Interlace-Bildschirm geschlossen wurde, kehrt die gesamte Anzeige wieder zum normalen Modus zurück. Ihr Fernsehschirm mag es eventuell nicht, wenn Sie oft und schnell zwischen dem normalen und dem Interlace-Modus hin- und herschalten. Deshalb sollten Sie damit etwas sparsam umgehen.

Auch bei Interlace-Bildschirmen sind alle normalen Funktionen wie Screen Offset, Screen Display usw. verfügbar. Das einzige Interlacing-Problem ergibt sich in Verbindung mit der Copperliste. Die Bitplanes werden während des Vertical Blanks verändert, und dieser spezielle Interlace-Prozeß ist bei der Berechnung der Copperliste verboten.

Wenn Sie also eine große Copperliste haben (z.B. 4 Bildschirme, 1 Interlace und einen Regenbogen) und Sie müssen eine Copper-Berechnung durchführen, so zeigt der Interlace-Bildschirm während der Berechnung nur das halbe Bild an. Dieses Problem läßt sich nicht umgehen, das System ist einfach in dieser Beziehung eingeschränkt.

HAM - Hold and Modify - Modus

Sie sind derzeit durch die Software des Amiga auf maximal sechs Bit-Planes pro Schirm beschränkt. Damit können Sie bis zu 64 verschiedene Farben gleichzeitig auf dem Schirm anzeigen. Wenn Sie aber jetzt zum Beispiel eine Photographie auf dem Schirm darstellen möchten, würden Sie Hunderte oder sogar Tausende von Farben auf dem Bildschirm benötigen.

Und genau das war das Problem, mit dem Jay Miner kämpfte, als er das Displaysystem des Amiga entwickelte. Er löste es mit einem Trick, den Maler schon seit Jahrhunderten kennen.

Wenn ein Maler jede erdenkliche Farbe zu einer Sitzung mitnehmen wollte, dann wäre das so gut wie unmöglich. Deshalb ist es allgemein üblich, die genaue Farbschattierung vor Ort aus einer kleinen Reihe von Grundfarben zu mischen. So kann man unzählige verschiedene Schattierungen erzeugen und muß nicht ganze Wagenladungen an Farbe mit sich herumschleppen.

Dieselbe Technik kann man auch auf einen Computer-Bildschirm anwenden.

Anstatt jede Farbe einzeln zu spezifizieren, können Sie eine vorhandene Farbe nehmen und sie etwas verändern. So wird wie Anzahl der verfügbaren Farben drastisch erhöht und die Grundlage des leistungsstarken Amiga HAM-(**H**old **A**nd **M**odify) Modus geschaffen.

Jeder Farbwert auf dem Amiga entsteht aus der Mischung von drei Bestandteilen. Sie bestimmen die relative Stärke der Grundfarben Rot, Grün und Blau in der Endfarbe. Die Farbintensität liegt dabei im Bereich zwischen 0 und 15.

Der HAM-Modus teilt die Farbwerte des Amiga in vier verschiedene Gruppen ein:

- **Farbregister 0-15:** Der Wert der ersten 16 Farben stammt direkt aus einem Farbregister. Diese Farben werden wie die 16 Farben auf einem Standardbildschirm behandelt.

- **Rote Komponenten 16-31:** Wenn für einen Punkt jedoch ein Wert im Bereich von 16 bis 31 festgelegt ist, wird der Farbwert von dem Pixel, das direkt links neben ihm liegt, geladen. Die rote Komponente dieser Farbe wird nun durch einen Wert von 0 bis 15 ersetzt, der sich nach folgender Formel errechnet:

  Intensität=Farbindex-16

- **Grüne Komponenten 32-47:** Entsprechend greift eine Farbnummer von 32 bis 47 auf die aktuelle Schattierung zu und verändert die grüne Komponente. Die Intensität dieser Komponente wird auf den Wert Farbe-32 festgelegt.

- **Blaue Komponenten 48-63:** Diese Farbnummern erfassen den Farbwert des Punktes links vom aktuellen Pixel und laden eine neue blaue Komponente folgendermaßen:

  Intensität=Farbindex-48

Die Farbe eines bestimmten Punktes hängt daher von den Farben aller links von ihm liegenden Punkte ab. So können Sie die feinen Abstufungen erzielen, die ideal für die Darstellung von Hautfarben sind. Sie können jedoch die Farbe jedes einzelnen Punktes auf dem Schirm nicht gesondert bestimmen. In der Praxis sind mindestens drei Pixel erforderlich, um von einer Farbe zu einer anderen überzugehen.

Als der Amiga auf den Markt kam, wurde HAM erst einmal nur als Kuriosität betrachtet. Inzwischen hat sich die Situation aufgrund der Entwicklung von so ausgezeichneten HAM-Grafik-Paketen wie Photon Paint völlig gewandelt.

Mit AMOS können Sie gesamte Spannweite der Grafik- und Text-Operationen direkt auf einem HAM-Bildschirm durchführen. **BEISPIEL 10.1** zeigt Ihnen an einem einfachen Beispiel, wie Sie mit nur ein paar Basic-Zeilen einen gesamten Bildschirm erstellen können.

Ein weiterer wichtiger Punkt ist, daß HAM-Schirme mit den normalen Befehlen SCREEN DISPLAY und SCREEN OFFSET manipuliert werden können. Hier sind ein paar einfache Richtlinien für ihren Einsatz:

- Für den ersten Punkt in jeder horizontalen Zeile sollte ein Farbwert von 0 bis 15 eingegeben werden. Er dient dann als Ausgangspunkt für alle Schattierung in der aktuellen Zeile.

- Kommen Sie bitte nicht auf die Idee, Ihre HAM-Schirme horizontal zu scrollen. Wenn Sie das nämlich versuchen, erhalten Sie an den Seiten Ihres Bildes einen Farbrand. Er entsteht dadurch, daß die Farbe des Anfangspunkts jeder Zeile nun verändert wird. Für das vertikale Scrollen gibt es jedoch keine Beschränkungen.

- Farbränder können auch durch den Befehl SCREEN COPY hervorgerufen werden. Die Lösung ist hier, sicherzustellen, daß die Umrandung Ihrer Zone in einer Farbe aus dem Bereich von 0 bis 15 gezeichnet wird. Dann werden Ihre HAM-Schirme an der neuen Position wieder in ihren Originalfarben dargestellt.

Das Laden eines Schirms

LOAD IFF *(Lädt einen IFF-Schirm von der Diskette)*

LOAD IFF "Dateiname"[,Schirm]

LOAD IFF lädt ein Bild im IFF-Format von der Diskette. Das IFF-Format wird jetzt von der überwiegenden Mehrheit der Grafik-Pakete für den Amiga unterstützt, deshalb sollte es Ihnen nicht schwerfallen, Ihre eigenen Kunstwerke direkt in AMOS-Basic zu laden.

Schirm gibt die Nummer des Bildschirms an, in den Sie ihr Bild laden möchten. Dieser Schirm wird automatisch für Sie eröffnet. Sollte er schon existieren, wird sein gesamter Inhalt vollständig gelöscht.

Wenn Sie das Bild in den aktuellen Schirm laden möchten, dann lassen Sie einfach den Parameter screen weg. Hier ist ein Beispiel:

```
Load Iff"AMOS_DATA:IFF/AMOSPIC.IFF",1
```

Das Speichern eines Schirms

SAVE IFF *(Speichert einen IFF-Schirm)*

SAVE IFF "Dateiname"[,Kompression]

SAVE IFF speichert den aktuellen Schirm als IFF-Bilddatei auf der Diskette. Kompression ist eine Markierung, mit der Sie wählen können, ob Ihre Datei vor dem Speichern komprimiert werden soll. Ein Wert von eins gibt an, daß das Standardsystem zum Komprimieren von Dateien angewandt werden soll, und bei einem Wert von Null wird das Bild unverändert gespeichert. Der Default für alle AMOS-Schirme ist auf komprimiert eingestellt.

SAVE IFF hängt automatisch ein kleines IFF-"Stück" an Ihre Bilddatei an. Darin werden die gegenwärtigen Bildschirm-Einstellungen einschließlich SCREEN DISPLAY, SCREEN OFFSET und SCREEN HIDE/SHOW aufbewahrt. Wenn Sie die Datei dann wieder in AMOS-Basic laden, erscheint sie haargenau in ihrem Originalzustand. Diese zusätzlichen IFF-Daten werden von externen Grafik-Paketen wie DPaint 3 völlig ignoriert.

Beachten Sie bitte, daß Sie Schirme, bei denen Sie die Double-Buffering- oder Dual- Playfield-Techniken eingesetzt haben, mit diesem Befehl nicht speichern können.

Bewegen eines Bildschirms

SCREEN DISPLAY *(Positionieren eines Bildschirms)*

SCREEN DISPLAY N [, x, y, w,h]

Nachdem Sie Ihre Bildschirme mit SCREEN OPEN definiert haben, können Sie sie auf Ihrem Schirm positionieren. Anders als die meisten Computer kann der Amiga ein Bild an jeder beliebigen Stelle auf dem Fernsehschirm darstellen. Das kann man gut nutzen, um die begeisternden Hüpf - Effekte hervorzurufen. In AMOS- Basic kann man diese Animationen sogar durch Interrupts ausführen (siehe AMAL).

Eine weitere Anwendung besteht darin, mehrere Schirme überlagert anzuordnen. Auf diese Art können Sie Ihr Display aus einer Kombination verschiedener Schirm- Modi zusammenstellen.

n gibt die Nummer des Bildschirms an, der positioniert werden soll. x und y stellen die Position des Schirms in Hardware-Koordinaten dar. Die x-Koordinate eines Schirms kann im Bereich von 0 bis 448 liegen und wird automatisch auf die nächstliegende 16-Pixel-Grenze gerundet. Auf Ihrem Fernsehschirm sind allerdings nur die Positionen zwischen 112 und 448 tatsächlich sichtbar, und wir würden Ihnen deshalb dringend raten, keine x-Koordinate unter 112 einzugeben.

Die y-Koordinate Ihres Schirms kann im Bereich von 0 bis 312 liegen. Der sichtbare Bereich hängt vor allem von Ihrem Fernsehschirm oder Monitor ab. Sie werden jedoch wahrscheinlich feststellen, daß Koordinaten, die zwischen 30 und 300 liegen, für die Mehrheit der Systeme gut geeignet sind.

Als dieses Handbuch geschrieben wurde, schien es im Amiga HAM-Modus noch einen kleinen Bug zu geben. Die Bilder können nämlich bei einer y-Koordinate von genau 256 nicht angezeigt werden. Sie müssen Ihre Bilder also stattdessen auf den benachbarten Koordinaten 255 oder 257 positionieren. Wir sind noch nicht ganz sicher, ob es sich hier um einen Hardware- oder Software-Fehler handelt, aber eigentlich werden Sie dadurch überhaupt nicht eingeschränkt.

w gibt die Breite Ihres Schirms in Pixel an. Wenn sie sich von den ursprünglichen Angaben unterscheidet, wird nur ein Teil ihres Bildes angezeigt, angefangen bei der linken oberen Ecke. Wie die x-Koordinate wird auch die Bildschirmbreite auf die nächstliegende 16-Pixel-Grenze gerundet.

h stellt entsprechend die Höhe Ihres Schirms dar. Eine Veränderung dieses Wertes führt zu einer Verringerung der Tiefe Ihres Bildes.

Im allgemeinen wählt SCREEN OPEN für Sie automatisch die Anzeigeposition anhand einer Standardeinstellung in der AMOS Konfigurationsdatei. Wenn ein Schirm größer als das Display ist, stellt AMOS den Schirm auf Overscan.

SCREEN DISPLAY bietet Ihnen also eine einfache Möglichkeit, die im Default gesetzten Werte zu verändern. Sie können jeweils jeden der Parameter *x,y,h* und *w* auch weglassen. Die Werte, die Sie hier nicht neu eingeben, sollten Sie durch Kommas voneinander abtrennen. Ihnen werden automatisch wieder die Default- Werte zugewiesen.

Screen Display 0,112,45,,: Rem Positioniert den Schirm bei 112,42

Wenn Sie Ihre Schirme positionieren, versuchen Sie immer sicherzugehen, daß der Schirm links oben am Display anfängt und dann nach rechts geht. Das ist unbedingt notwendig, denn sonst kann die Amiga Hardware Ihren Schirm nicht richtig umsetzen. In der Praxis werden Sie wohl hier etwas experimentieren müssen, bis Sie genau den gewünschten Effekt erzielen. Glücklicherweise kann Ihnen dabei nichts Schlimmeres passieren, als daß Ihr Display vielleicht etwas komisch aussieht. Wenn Sie hier einen Fehler machen, stürzt der Amiga noch lange nicht ab. Wir wollen Ihnen mit den folgenden Hinweisen ein bißchen weiterhelfen:

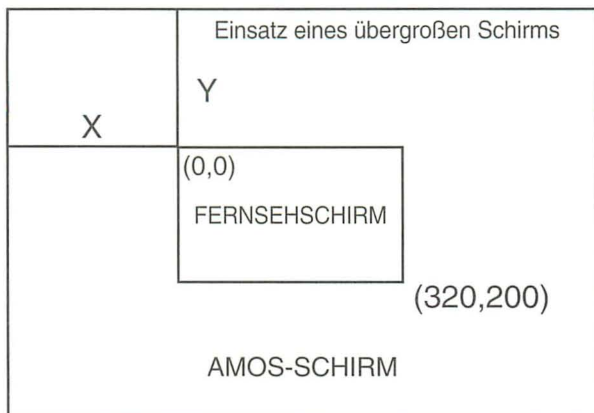
- Auf jeder horizontalen Zeile kann nur ein einziger Bildschirm angezeigt werden. Sie können jedoch mehrere Bildschirme aufeinander stapeln. Vorausgesetzt, es ist immer nur einer der Schirme sichtbar, geht dabei alles klar.

- Zwischen zwei Schirmen ist stets eine "tote Zone", die einen Pixel breit ist. Sie wird von der Copperliste hervorgerufen und ist absolut unumgänglich. Sie werden die tote Zone bemerken, wenn Sie ein Sprite zwischen den Schirmen bewegen. Gehen Sie doch zum Beispiel einmal mit dem Mauszeiger vom Editier- Fenster zur Menüzeile. An der Grenze der beiden Schirme sehen Sie jetzt eine dünne Linie, die durch Ihren Mauszeiger läuft.

SCREEN OFFSET *(Hardware-Scrollen)*

SCREEN OFFSET n, x, y

Das Amiga-Display ist nicht nur auf die sichtbaren Abmessungen Ihres Fernsehschirms begrenzt. Sie können nämlich Bilder erstellen, die wesentlich größer sind als der tatsächlich sichtbare Schirm. Es ist natürlich nicht möglich, solche Bilder dann in voller Größe darzustellen, aber Sie können mit dem Befehl SCREEN OFFSET ganz leicht einen Teil Ihres Bildes ansehen. Das wollen wir an folgendem Diagramm verdeutlichen:



Wie Sie sehen, wird der bezeichnete Bildschirmbereich durch ein Sichtfenster an den Koordinaten x und y angezeigt. Wenn Sie die Werte dieser Koordinaten entsprechend ändern, können Sie dieses Sichtfenster über den ganzen Schirm bewegen. So entsteht ein glatter Hardware-Scroll-Effekt, der für viele Spiele ganz ideal ist. Die Größe Ihres Sichtfensters beruht auf den Abmessungen, die Sie zuvor mit dem Befehl SCREEN DISPLAY festgelegt haben.

n ist die Nummer des Schirms, den Sie anzeigen möchten. *x,y* geben den Offset von der linken oberen Ecke des Schirms bis zum Ausgangspunkt Ihres Displays an. *x* und *y* werden in Einheiten von einem Pixel gemessen, also können Sie ohne weiteres wunderbar glattes Scrollen erreichen.

Sie können bei dieser Anweisung auch negative Abstände eingeben, und so jeden beliebigen Bereich des Amiga-Speichers auf dem Schirm darstellen. Eine umfassende Demonstration dieses Befehls finden Sie in **BEISPIEL 10.2**.

Befehle zur Bildschirmsteuerung

SCREEN CLONE *(Klont einen Schirm)*

SCREEN CLONE n

Der Befehl SCREEN CLONE weist dem Schirm mit der Nummer n eine zweite Version des aktuellen Schirms zu. Dieser Klon greift auf genau denselben Speicherbereich zu wie der ursprüngliche Bildschirm.

Normalerweise wird der geklonte Schirm an derselben Stelle wie der Originalschirm dargestellt. Er kann jedoch unabhängig mit den normalen Schirm-Befehlen wie SCREEN DISPLAY und SCREEN OFFSET manipuliert werden.

Da sich nur eine **einzige** Version der ursprünglichen Schirmdaten im Speicher

befindet, können Sie mit dem SCREEN-Befehl auf den Klon nicht zugreifen. Wenn Sie es trotzdem versuchen, erhalten Sie die Fehlermeldung Falsche Screen Parameter. Des weiteren sollten Sie bedenken, daß die Flash-Sequenzen, die Sie für die Farben des ursprünglichen Bildschirms eingegeben haben, nicht durch das Klonen kopiert werden. Siehe dazu auch **BEISPIEL 10.3**. Beachten Sie den Einsatz des Befehls WAIT VBL. Dadurch wird der Klon außerhalb des Bildschirms neu positioniert und die Bewegungen laufen weiterhin fließend ab.

Wenn Sie mit SCREEN CLONE etwas herumexperimentieren, dann werden Sie schnell feststellen, daß die Anzahl der möglichen Bewegungen eingeschränkt ist. Überschreitet man dieses Limit, kann dies zu unerträglichen Störungen führen. So kann z.B. das Hinzufügen einer zusätzlichen Berechnung in die Bewegungsroutine zu langen Wartezeiten während der Bewegung führen.

Sie können die Bildschirmanzeige auch direkt aus der AMAL-Animationssprache heraus anpassen. Mit AMAL können Sie eine große Anzahl von Schirmen leicht und glatt animieren. **BEISPIEL 10.4** demonstriert Ihnen das sehr gut. Die Animationen laufen jetzt so schnell ab, daß Sie sie verlangsamen müssen, um sie genau zu sehen!

DUAL PLAYFIELD *(Kombiniert zwei Schirme zu einem Dual Playfield)*

DUAL PLAYFIELD Schirm1, Schirm2

Der Dual-Playfield-Modus des Amiga ermöglicht es Ihnen, zwei vollständige Schirme gleichzeitig an denselben x- und y-Koordinaten darzustellen. Das ist so, als würden Sie jeden der Schirme auf Cellophan zeichnen und sie übereinanderlegen. Jeder Schirm kann dabei total unabhängig vom anderen manipuliert werden. Sie können dies nützen, um einen glatten Parallax-Effekt hervorzurufen, der ideal für Scroll-Spiele wie Silkworm ist.

Die beiden Komponenten eines Dual Playfields werden wie jeder andere AMOS-Bildschirm behandelt und können auf ganz normale Art geschrieben werden. Sie können auch mit AMAL animiert oder mit Double Buffering versehen werden.

Schirm1 und Schirm2 beziehen sich auf Schirme, die Sie zuvor mit dem Befehl SCREEN OPEN eröffnet haben. Dabei sind jedoch nur bestimmte Bildschirmkombinationen zulässig. Beide Schirme müssen die gleiche Auflösung haben, denn es ist nicht möglich, Hires- und Lowres-Schirme in derselben Spielfläche zu verwenden.

Hier ist eine Liste der Möglichkeiten:

Schirm1	**Schirm2**	**Bemerkungen**
Anzahl d. Farben	Anzahl d. Farben	
2	2	
4	2	
4	4	
8	4	Nur Lowres
8	8	Nur Lowres

Die Farbbereiche werden zwar vorher definiert, aber die Größe der beiden Schirme kann völlig unterschiedlich sein. Sie können einen phantastisch realistischen Parallax-Effekt hervorrufen, indem Sie einen Hintergrundschirm erzeugen, der größer als der Vordergrund ist.

Alle Farben dieser Schirme stammen aus der Palette von screen1, wobei für die Farbe Null transparent eingesetzt wird.

Schirm	Farbindizes (von Schirm 1)
1	0-7
2	8-15

Wenn Sie auf dem zweiten Schirm zeichnen, konvertiert AMOS-Basic Ihren Farbindex vor dem Einsatz automatisch in den entsprechenden Wert. So wird für INK2 die Farbe mit der Nummer neun aus der ersten Palette eingesetzt.

Dieser Umwandlungsprozeß wird jedoch nicht auf die Zuweisungsbefehle wie COLOUR oder PALETTE angewandt. Das sollten Sie nicht vergessen, wenn Sie die Farbeinstellungen ändern, sonst werden Ihre neuen Farben nicht auf dem aktuellen Schirm erscheinen. Bevor Sie die Farbzuweisung ändern, machen Sie stets erst *screen1* zum aktuellen Schirm.

```
Screen 1 : Rem Wenn Schirm  1 die Nummer des ersten Schirms ist
```

Es gibt einige wichtige Punkte, die Sie beachten sollten, bevor Sie ein Dual Playfield erzeugen:

- Die Schirm-Offsets für beide Schirme dürfen nie auf Null gesetzt werden.
- Wenn Sie einen Dual-Playfield-Schirm erzeugt haben und ihn nun mit SCREEN OFFSET bewegen möchten, müssen Sie Schirm 1 und nicht 2 angeben.

DUAL PLAYFIELD ist eine äußerst leistungsstarke Anweisung. Sie erhalten in **BEISPIEL 10.5** eine vollständige Demonstration.

DUAL PRIORITY *(Bestimmt die Reihenfolge von Dual-Playfield-Schirmen)*

DUAL PRIORITY Schirm1, Schirm2

Normalerweise wird der erste Schirm eines Dual Playfields direkt über dem zweiten angezeigt. Mit dem Befehl DUAL PRIORITY können Sie diese Reihenfolge verändern und *screen2* erscheint dann vor *screen1*.

Achtung: Diese Anweisung verändert nur die Reihenfolge der Anzeige. Sie hat **keine** Wirkung auf die Anordnung der Bildschirme. Sie sollten trotzdem den ersten Schirm in der Dual-Playfield-Liste für alle Farbzuweisungen und den Einsatz mit dem Befehl SCREEN OFFSET verwenden.

SCREEN *(Bestimmt aktuellen Schirm)*

SCREEN n

Mit dem SCREEN-Befehl können Sie alle Grafik- und Textoperationen auf den Schirm mit der Nummer *n* leiten. Wenn dieser Schirm versteckt ist oder sich außerhalb des Anzeigebereichs befindet, hat dieser Befehl keine sichtbare Wirkung. Die Grafiken werden jedoch trotzdem im Speicher erstellt und angezeigt, sobald der Schirm im Blickfeld erscheint.

=SCREEN *(Zeigt die aktuelle Bildschirm-Nummer an)*

s=SCREEN

Weist die Nummer des gegenwärtig aktiven Bildschirms aus. In diesem Schirm finden alle Zeichenschritte statt, aber er ist **nicht** unbedingt auch sichtbar.

SCREEN TO FRONT *(Bringt den Schirm in den Vordergrund des Displays)*

SCREEN TO FRONT [s]

Mit dieser Anweisung wird der Schirm *s* in den Vordergrund des Fernsehschirms gerückt. Wenn Sie den Parameter *s* nicht eingeben, so wirkt diese Anweisung stattdessen auf den aktuellen Schirm.

```
Screen Close 0 : Load Iff"AMOS_DATA:IFF/AMOSPIC.IFF",0
Load Iff "AMOS_DATA:IFF/Magic_Forest.IFF",1
Wait Key : Screen To Front 0
```

Achtung: Wenn Sie das AUTOVIEW-System abgeschaltet haben, müssen Sie den VIEW-Befehl aufrufen, bevor die Wirkung auf dem Schirm sichtbar wird.

SCREEN TO BACK *(Schiebt den Bildschirm in den Hintergrund)*

SCREEN TO BACK[n]

SCREEN TO BACK schiebt einen Bildschirm in den Hintergrund Ihres Displays. Wenn an denselben Koordinaten ein weiterer Bildschirm positioniert ist, dann wird dieser jetzt im Vordergrund des entsprechenden Bildschirms angezeigt.

SCREEN HIDE *(Versteckt einen Schirm zeitweise)*

SCREEN HIDE [n]

Entfernt den bezeichneten Bildschirm völlig aus dem Blickfeld. Dieser Schirm kann durch das Aufrufen von SCREEN SHOW wieder angezeigt werden. Wenn Sie für n nichts angeben, bezieht sich diese Anweisung auf den aktuellen Schirm.

SCREEN SHOW *(Zeigt einen Schirm wieder an)*

SCREEN SHOW n

Mit SCREEN SHOW machen Sie einen Schirm, den Sie zuvor mit SCREEN HIDE versteckt haben, wieder sichtbar.

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",1
Screen Hide : Wait Key : Screen Show
```

=SCREEN HEIGHT *(Gibt die Höhe des Bildschirms an)*

h=SCREEN HEIGHT [n]

Gibt die Höhe eines AMOS-Schirms an. Wenn Sie bei dieser Anweisung den Parameter n weglassen, bezieht sie sich automatisch auf den aktuellen Bildschirm.

```
Print Screen Height
```

=SCREEN WIDTH *(Gibt die Breite des Bildschirms an)*

w=SCREEN WIDTH [n]

SCREEN WIDTH weist entweder die Breite des aktuellen oder des Schirms mit der Nummer *n* aus.

```
Print Screen Width
```

=SCREEN COLOUR *(Gibt die Anzahl der Farben an)*

c=SCREEN COLOUR

Die Anweisung SCREEN COLOUR gibt die maximale Anzahl der Farben auf dem gegenwärtig aktiven Bildschirm an.

```
Print Screen Colour
```

=SCIN *(Gibt die Nummer des Bildschirms an einer bestimmten Position an)*

s=SCIN(x,y)

Mit diesem Befehl erhalten Sie die Nummer des Bildschirms, der bei den **HARDWARE**-Koordinaten *x,y* liegt. Wenn dort kein Schirm existiert, wird s mit einem negativen Wert (Null) belegt.

SCIN wird normalerweise in Verbindung mit den Funktionen X MOUSE und Y MOUSE eingesetzt, um festzustellen, ob der Maus-Cursor in einen bestimmten Schirm eingetreten ist. Hier ist ein Beispiel:

```
Print Scin(X Mouse,Y Mouse)
```

Das Definieren der Bildschirmfarben

DEFAULT PALETTE *(Lädt den Bildschirm mit Standardpalette)*

DEFAULT PALETTE c1,c2,c3,,,c6,,,- bis zu 32 Farben

Durch diesen Befehl wird das Eröffnen von vielen Schirmen mit derselben Palette vereinfacht. So definiert man eine Reihe von Farben, die dann für sämtliche Schirme, die Sie danach mit SCREEN OPEN eröffnen, verwendet werden. Wie üblich liegen die zulässigen Farbwerte im Bereich zwischen $000 und $FFF ($RGB). Siehe auch GET SPRITE PALETTE.

GET PALETTE *(Bestimmt die Palette eines Schirms)*

GET PALETTE n [,Maske]

Die Anweisung GET PALETTE kopiert die Farben des Schirms n und lädt sie in den aktuellen Schirm. Das kann sehr nützlich sein, wenn Sie mit SCREEN COPY Informationen von einem Schirm auf einen anderen übertragen. Es ist nämlich oft unbedingt erforderlich, daß sowohl der Quell- wie auch der Zielschirm über dieselben Farbeinstellungen verfügen.

Wahlweise können Sie über den *Masken*-Parameter auch nur eine Auswahl der Farben laden. Unter GET SPRITE PALETTE finden Sie die Einzelheiten zu diesem Parameter. Hier nun ein Beispiel:

```
Load iff "AMOS_DATA:Iff/AMOSPIC.IFF",0
Screen Open 1,Screen Width, Screen Height, Screen Colour, Lowres
Screen Copy 0,0,0,160,100 To 1,80,80
Centre "<Taste drücken und Palette holen>" : Wait Key
Get Palette 0
```

Das Löschen des Schirms

CLS *(Löscht den Schirm)*

Mit CLS können Sie den aktuellen Schirm teilweise oder ganz löschen. Für diesen Befehl gibt es drei mögliche Formate:

CLS

Löscht den aktuellen Schirm durch das Ausfüllen mit der Farbe Null. Außerdem werden durch diese Anweisung alle Fenster, die eröffnet wurden, gelöscht.
CLS col

Füllt Ihren Schirm mit der Farbe *col.*

CLS col,x1,y1 to x2,y2

Ersetzt den rechteckigen Bereich an den Koordinaten *x1,y1,x2,y2* mit einem Block der Farbe *col.* Für col kann jeder Wert von 0 bis zur Höchstanzahl der verfügbaren Farben eingegeben werden.

x1,y1,x2,y2 stellen die Koordinaten der linken oberen und der rechten unteren Ecke des Bereichs dar, der durch diesen Befehl gelöscht werden soll. Hier ist ein Beispiel:

```
Cls : Circle 100,98,98 : Cls 1,50,50 To 150,150
```

Das Manipulieren des Schirminhaltes

SCREEN COPY *(Kopiert Teile des Schirms)*

SCREEN COPY scr1 TO scr2

SCREEN COPY scr1,x1,y1,x2,y2 TO scr2,x3,y3[,Modus]

Mit dem Befehl SCREEN COPY können Sie große Bereiche eines Schirms mit atemberaubender Geschwindigkeit von einer Stelle an eine andere kopieren.

scr1 stellt dabei den Schirm dar, der als Quelle für Ihr Bild dient. Hier können Sie entweder eine ganz normale Schirm-Nummer oder die Nummer eines logischen oder physischen Schirms eingeben, den Sie mit den LOGIC- oder PHYSIC-Befehlen erzeugt haben.

Mit *scr2* kann man wahlweise einen Schirm angegeben, in den diese Daten kopiert werden. Wenn Sie diesen Parameter weglassen, wird der Bereich in den aktuellen Schirm kopiert.

x1,y1 und *x2,y2* geben die Abmessungen eines rechteckigen Quellbereichs an, und

x3,y3 stellen die Koordinaten des Zielbereichs dar. Diese Koordinaten unterliegen keinerlei Beschränkungen. Alle Teile Ihres Bildes, die außerhalb Ihres aktuellen Schirmbereichs liegen, werden automatisch entsprechend abgeschnitten.

In **BEISPIEL 10.6** im Manual-Unterverzeichnis finden Sie ein Beispiel für den Einsatz des Befehls SCREEN COPY.

Über den Parameter *Modus* können Sie wahlweise eingeben, welcher der 255 möglichen Blitter-Modi für Ihren Kopiervorgang eingesetzt werden soll. Dieser Modus bestimmt, wie Ihre Quell- und Zielbereiche auf dem Schirm miteinander verbunden werden. Der Modus wird in folgendem Format über ein Bitmuster eingeben:

Modus-Bit	Quell-Bit	Ziel-Bit
4	0	0
5	0	1
6	1	0
7	1	1

Beachten Sie hier bitte, daß die unteren vier Bits in diesem Muster nicht von dieser Anweisung verwendet werden und daher immer auf Null gestellt werden sollen.

Jedes Bit in *mode* stellt eine Bit-Kombination in Quell- und Zielbereich dar. Wenn ein Modus-Bit auf eins gestellt ist, dann wird das entsprechende Bit auf dem Schirm auch mit eins geladen, andernfalls ist das Ergebnis Null.

Um den richtigen Zeichen-Modus für Ihre Anwendung zu finden, müssen Sie einfach nur entscheiden, welche Kombinationen eins ergeben sollen und die entsprechenden Bits im *mode*-Parameter daraufhin einstellen.

Angenommen, Sie möchten nur dann ein Bit auf dem Schirm einstellen, wenn Quell- und Zielbit übereinstimmen. Jetzt suchen Sie in der Tabelle nach den Punkten, an denen Ihre Bedingung erfüllt wird. Auf diese Art erhalten Sie den folgenden Wert für *mode*:

%10010000

Wenn Sie mit Binärangaben nicht vertraut sind, erscheint Ihnen dieser Befehl vielleicht etwas undurchsichtig. Anstatt Sie jetzt tödlich mit einer Erklärung für "binär" zu langweilen, geben wir Ihnen stattdessen eine ausführliche Auflistung der häufigsten Anforderungen mit den zugehörigen Bit-Maps.

Modus	Wirkung	Bitmuster
REPLACE	Ersetzt das Ziel durch eine direkte Kopie des Quellbildes (Default).	%11000000
INVERT	Ersetzt das Zielbild durch eine inverse Kopie des Quellbildes.	%00110000
AND	Kombiniert Quelle und Ziel durch eine logische AND-Operation.	%10000000

Modus	Wirkung	Bitmuster
OR	Verbindet die Quelle mit dem Zielbild durch OR.	%11100000
XOR	Verbindet Quell- und Zielbereich durch Exclusive OR.	%01100000

Technisch orientierte Anwender sollten beachten, daß SCREEN COPY Quelle und Ziel unter Verwendung der Blitter-Bereiche B und C verbindet. Blitter-Bereich A wird von dem System überhaupt nicht eingesetzt.

Das Scrollen des Schirms

DEF SCROLL *(Definiert eine Scroll-Zone)*

DEF SCROLL n,x1,y1 to x2,y2,dx,dy

DEF SCROLL ermöglicht Ihnen die Definition von bis zu 16 verschiedenen Scroll-Zonen. Jede dieser Zonen kann mit einer bestimmten Scroll-Operation belegt werden, die durch die Variablen *dx* und *dy* bestimmt werden.

n stellt dabei die Nummer der Zone dar und kann im Bereich zwischen 1 und 16 liegen. *x1,y1* beziehen sich auf die Koordinaten der linken oberen Ecke des Bereichs, der gescrollt werden soll, und *x2,y2* bezeichnen die Koordinaten der diagonal gegenüberliegenden Ecke.

dx stellt die Anzahl der Pixel der Zone dar, die bei jedem Schritt nach rechts geschoben werden. Dabei weisen negative Werte darauf hin, daß das Scrollen von rechts nach links erfolgt, und positive Werte zeigen an, daß von links nach rechts gescrollt wird.

Entsprechend gibt *dy* die Anzahl der Punkte an, um die die Zone nach oben oder unten gescrollt wird. Hier zeigen negative Werte eine Bewegung nach oben, und positive Werte ein Verschieben nach unten an.

SCROLL *(Scrollt den Schirm)*

SCROLL n

Der SCROLL-Befehl scrollt den Schirm anhand der Einstellungen, die Sie mit der Anweisung DEF SCROLL eingegeben haben. *n* bezeichnet die Zone, die Sie scrollen möchten.

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",2
Def Scroll 1,0,0 to 320,200,1,0
Do
```

```
    Scroll 1
  Loop
```

In **BEISPIEL 10.7** und **BEISPIEL 10.8** können Sie noch ausführlichere Beispiele finden. Hier wird ein Bild von der AMOS System-Diskette geladen und auf dem Schirm rotiert. Die Variable S enthält die Anzahl der Punkte des Bildes, die bei jedem Scrollen bewegt werden. Je größer der Wert von S ist, desto schneller und ruckartiger erfolgt das Scrollen. Beachten Sie, wie hier die Qualität der Bewegung durch das Austauschen der Schirme sichtbar verbessert wird.

Das Austauschen der Schirme

Um den glatten Bewegungsablauf hervorzurufen, den Sie in einem Computer-Spiel erwarten, müssen alle Zeichenschritte in einer Zeitspanne von unter einem Fünfzigstel einer Sekunde ausgeführt werden. Das fordert selbst den schnellsten Computer auf äußerste, und auch auf einem Amiga schaft man es oft nicht. Es ist äußerst ärgerlich, aber bei einer komplexen Animation flimmern daher Ihre Grafiken ständig, während sie gezeichnet werden.

Glücklicherweise gibt es für dieses Problem eine Lösung, die erfolgreich bei der Mehrheit der Spielhallen-Spiele eingesetzt wird. Man nennt diese Technik "Austauschen der Schirme" (screen switching). Auf diese Art kann man leicht mit nur einem Bruchteil der Amiga-Kapazität glatte, saubere Animation erzeugen.

Das Grundprinzip ist ziemlich einfach. Anstatt Ihre Bilder auf dem aktuellen Schirm zu erstellen, führen Sie alle Zeichenschritte auf einem besonderem *logischen* Schirm aus, der für den Anwender völlig unsichtbar ist. Dieser Schirm unterscheidet sich völlig vom physischen Schirm, der gerade auf Ihrem Fernsehschirm angezeigt wird. Wenn Sie die Grafiken dann fertiggestellt haben, können Sie den logischen und den physischen Schirm vertauschen, um einen reibungslosen Übergang zwischen den beiden Schirmbildern zu erreichen. So wird der vorherige *physische* Schirm jetzt zum neuen logischen Schirm, und auf ihm erstellen Sie nun das nächste Bild in Ihrer Sequenz.

Auf dem ersten Blick sieht die ganze Geschichte ziemlich kompliziert aus, aber das alles wird durch den AMOS-Befehl DOUBLE BUFFER automatisch durchgeführt. Durch diesen Befehl können alle Zeichenschritte nur direkt auf dem logischen Schirm ohne Auswirkungen auf die aktuelle Anzeige durchgeführt werden. Sie müssen in Ihrem Programm nur noch die Zeichenoperationen mit dem Austauschen der Schirm synchron schalten. Und das geht mit Hilfe der Anweisung SCREEN SWAP.

SCREEN SWAP *(Vertauscht den logischen und physischen Schirm)*

SCREEN SWAP [n]

Durch den Befehl SCREEN SWAP werden der logische und der physische Schirm miteinander vertauscht. Auf diese Weise können Sie die physische Anzeige der beiden Schirme ohne Verzögerung vertauschen.

Wenn Sie den Befehl DOUBLE BUFFER eingesetzt haben, sind die beiden Schirme bereits für Sie erzeugt worden. Sie müssen aber noch das automatische Austausch-

System mit dem Befehl BOB UPDATE OFF ausschalten, sonst werden die Schirme fünfzigmal pro Sekunde ausgetauscht. Und das würde Sie bestimmt bei Ihren Zeichenoperationen stören. Außerdem müssen Sie auch die Funktion AUTOBACK OFF ausschalten. Mit dieser Funktion werden normalerweise Ihre Grafikoperationen sowohl auf den physischen wie auch auf den logischen Schirm kopiert. Sie ist ganz nützlich, wenn Sie einfache Grafiken mit animierten BOBs verbinden wollen, macht aber andererseits die durch das Austauschen der Schirme erzeugte Wirkung völlig zunichte.

Wie leistungsstark dieser Befehl ist, können Sie anhand der Programme in **BEISPIEL 10.9** und **BEISPIEL 10.10** sehen.

In **BEISPIEL 10.9** wird ein Dreieck vollkommen ohne den Einsatz der Screen-Switching-Technik über den Schirm bewegt. Während es sich bewegt, wird das Flimmern zunehmend stärker und störender. Jetzt wollen wir dieses Programm durch das Austauschen der Schirme etwas aufmöbeln. Wir zeichnen dazu das Dreieck auf einen unsichtbaren logischen Schirm und bringen es ins Blickfeld, wenn es ganz fertig ist. Das neue Programm finden Sie jetzt in **BEISPIEL 10.10**.

In diesem Beispiel wird jedes neue Dreieck ohne Auswirkungen auf das aktuelle Display auf den Schirm gezeichnet. Die Anweisung SCREEN SWAP vertauscht dann den logischen und den physischen Schirm, so daß die fertige Version des Dreiecks sofort auf dem Schirm erscheint - ohne das geringste Flackern. Dann wird das alte Dreieck vom logischen Schirm gelöscht und an der nächsten Position neu gezeichnet. Wenn das Programm nun abläuft, läuft das Dreieck in einer fließenden Bewegung von einer Ecke zur anderen.

Beachten Sie bitte, daß wir das Flackern in **BEISPIEL 10.9** absichtlich übertrieben dargestellt haben, um die Technik des Austauschens der Schirme zu veranschaulichen. In der Praxis könnte man das Problem auch ohne den Einsatz der Anweisung SCREEN SWAP noch beträchtlich abschwächen.

=LOGBASE *(Gibt die Adresse eines Teils des logischen Schirms an)*

Adresse=LOGBASE(Plane)

Die Funktion LOGBASE zielt auf die erfahrenen Programmierer ab, die direkten Zugang zum Bildschirmspeicher des Amiga erhalten möchten.

plane bezieht sich auf eine von sechs möglichen Bitplanes, aus denen sich der aktuelle Schirm zusammensetzt. Nach dem Aufrufen von LOGBASE erscheint in *adresse* entweder die Adresse der betreffenden Bitplane oder - wenn sie nicht existiert - Null.

=PHYBASE *(Gibt die Adresse des aktuellen Speichers an)*

Adresse=PHYSBASE (Plane)

Durch den Befehl PHYBASE erhalten Sie die Speicheradresse einer Bitplane mit der Nummer *plane* für den aktuellen Schirm. Diese Funktion weist einen Wert Null aus, wenn die *plane* nicht existiert. Hier ist ein Beispiel:

```
Loke Phybase(0),0 : Rem Pokt eine dünne Linie direkt auf den Schirm
```

=PHYSIC *(Ermöglicht die Identifikation des physischen Schirms)*

=PHYSIC
=PHYSIC(s)

Die PHYSIC-Funktion gibt eine Identifikationsnummer für den aktuellen physischen Schirm an. Über diese Nummer erhalten Sie direkten Zugang zum physischen Bild, das vom Double-Buffering-System angezeigt wird.

Das von dieser Funktion ausgewiesene Ergebnis kann die Schirmnummer bei den Befehlen ZOOM, APPEAR und SCREEN COPY ersetzen.

s ist die Nummer eines AMOS-Schirms. Wenn Sie sie weglassen, wird stattdessen der aktuelle Schirm verwendet. Verwechseln Sie diese Funktion bitte nicht mit LOGBASE.

=LOGIC *(Gibt die Identifikationsnummer des logischen Schirms an)*

=LOGIC
=LOGIC(s)

Mit dieser Anweisung erhält man eine Identifikationsnummer für einen logischen Schirm. Die Identifikationsnummer kann in Verbindung mit den Befehlen SCREEN COPY, APPEAR und ZOOM eingesetzt werden, um Ihr Bild außerhalb des Schirms ohne Auswirkungen auf das aktuelle Display zu verändern.

Das Synchronisieren des Bildschirms

Wie bei den meisten Heim-Computern beruht auch das Amiga-Display auf dem Prinzip einer Bildspeichertabelle. Dies ist nur der Fachausdruck für ein Konzept, mit dem Sie bestimmt schon vertraut sind. Einfach ausgedrückt, bedeutet die Anzeige auf der Basis einer Bildspeichertabelle, daß spezielle Hardware eingesetzt wird, um ein gespeichertes Bild in ein Signal umzusetzen, das dann auf Ihrem Fernsehschirm angezeigt werden kann. Also immer wenn AMOS-Basic auf den Schirm zugreift, geschieht das über diesen Bildschirmspeicher.

Die Anzeige auf dem Schirm wird von der Hardware alle Fünfzigstelsekunden aktualisiert. Nachdem ein Schirm gezeichnet wurde, schaltet sich der elektronische Strahl ab und kehrt nach links oben auf den Schirm zurück. Diesen Vorgang bezeichnet man als Vertical Blank Period oder VBL. Gleichzeitig führt AMOS-Basic eine Reihe von wichtigen Aufgaben durch, wie das Bewegen der Sprites und Anpassen der Adresse des physischen Schirms, wenn sie sich geändert hat. Die Ausführung von Anweisungen wie ANIM und SCREEN SWAP ist daher erst dann vollständig beendet, wenn der Schirm neu gezeichnet wird.

Da ein Fünfzigstel einer Sekunde für AMOS-Basic eine ziemlich lange Zeit ist, kann es zu einem ernsthaften Koordinationsproblem zwischen Ihrem Programm und dem Schirm kommen, die vor allem bei engen Programmschleifen unangenehm auffallen. Am besten können Sie diese Schwierigkeiten vermeiden, indem Sie warten, bis der Schirm aktualisiert wurde, bevor Sie den nächsten Basic-Befehl ausführen.

WAIT VBL *(Auf eine Vertical Blank Period warten)*

Mit der Anweisung WAIT VBL halten Sie den Amiga an, bis die nächste Vertical Blank Period auftritt. Es ist üblich, diese Anweisung entweder nach den Befehlen PUT BOB oder SCREEN SWAP einzusetzen.

Spezialeffekte

APPEAR *(Ausblenden zwischen zwei Bildern)*

APPEAR Quelle TO Ziel,Effect [,Pixel]

Der Befehl APPEAR ermöglicht Ihnen das raffinierte Ausblenden von *Quell-* und *Ziel-*Schirmen. Dabei geben Sie für *Quelle* und *Ziel* einfach die Nummern der Bildschirme ein, die Sie zuvor mit SCREEN OPEN eröffnet haben. Falls erforderlich können Sie hier auch die Funktionen LOGIC und PHYSIC einsetzen.

Mit *Effekt* bestimmen Sie, wie diese Anweisung ausgeführt werden soll. Die Größe dieses Parameters kann von sich 1 bis zur Anzahl der Pixel in Ihrem aktuellen Schirm bewegen.

Pixel gibt die Anzahl der Punkte an, auf die dieser Befehl wirkt. Normalerweise setzt man diesen Wert auf TOTAL, also auf den gesamten Bildschirmbereich, aber Sie können ihn auch verringern und nur einen Teil des Schirms ausblenden. Alle Schirme werden stets von oben nach unten gezeichnet.

Wie das Ausblenden letztendlich genau umgesetzt wird, hängt natürlich davon ab, in welchem Schirm-Modus Sie arbeiten. In **BEISPIEL 10.11** finden Sie ein Beispielprogramm. Hier können Sie nun selbst die verschiedenen Möglichkeiten ausprobieren.

FADE *(Mischen von einer oder mehreren Farben zu neuen Farbwerten)*

FADE Geschwindigkeit [,Farbenliste]
FADE Geschwindigkeit TO Schirm [,Maske]

Der FADE-Befehl ermöglicht es Ihnen, den glatten Übergang einer gesamten Palette von einem Satz Farben zu einem anderen zu erzeugen. Auf diese Art können Sie in Ihren Lade-Schirmen höchst professionelle Ausblendeffekte erzielen.

In der Standardversion dieser Anweisung wird die aktuelle Palette erfaßt und die Schirmfarben langsam bis auf Null aufgelöst. Dabei wird jeder Farbwert nach und nach um eins reduziert, bis er auf Null steht. Dazu folgendes Beispiel:

```
Fade 15:Wait 225
```

Geschwindigkeit gibt die Anzahl der Vertical Blank Periods (VBLs) an, die vor der nächsten Farbänderung auftreten müssen.

Da das Ausblenden mit Hilfe von Interrupts ausgeführt wird, warten Sie mit der Eingabe der nächsten Basic-Anweisung am besten, bis der Vorgang völlig abgeschlossen ist. Die Zeit, die für WAIT berücksichtigt werden soll, kann nach folgender Formel berechnet werden:

Wartezeit = Ausblendgeschwindigkeit * 15

Die Funktion von FADE kann auf das Erzeugen einer neuen Palette direkt aus einer Liste von Farbwerten ausgeweitet werden.

Fade 15,,$100,$200,$200,$300

Bei dieser Anweisung kann jede beliebige Anzahl von Farben angegeben werden. Die mögliche Höchstanzahl wird durch den jeweiligen Grafik-Modus, in dem Sie gerade arbeiten, bestimmt. Wie bei den meisten AMOS-Befehlen können Sie auch hier bestimmte Parameter völlig weglassen. Auf diese Farben hat der FADE-Befehl dann keine Auswirkung.

Fade 15,,$100,$800,$F00

In ihrer stärksten Ausführung kann die FADE-Anweisung einen fließenden Übergang der Farben des aktuellen Bildschirms zu einer Palette aus einem anderen vorhandenen Schirm bewirken.

FADE Geschwindigkeit TO s[,Maske]

Die gegenwärtigen Farben werden langsam in die Farbpalette des Schirms s umgewandelt. Mit derselben Technik können Sie hier auch die Palette aus der Sprite-Bank laden. Hier geben Sie einfach für die Bildschirmnummer s einen negativen Wert ein.

Maske ist ein Bitmuster, das angibt, welche Farben geladen werden sollen. Jeder Farbe entspricht ein Bit dieses Musters mit einer Nummer zwischen 1 und 15. Wenn ein Bit also auf 1 gestellt wird, so führt das zur Veränderung der entsprechenden Farbe. Siehe dazu auch **BEISPIEL 10.12**.

FLASH *(Stellt blinkende Farbsequenz ein)*

Mit diesem Befehl können Sie eine periodische Veränderung der Farbe, die jedem Farbindex zugewiesen wurde, bewirken. Das geschieht durch ein Interrupt, auf ähnliche Art wie bei den Sprite- und Musikanweisungen. Das Format des Blink- Befehls lautet folgendermaßen:

FLASH index,"(Farbe, Verzögerung)(Farbe, Verzögerung)(Farbe, Verzögerung)..."

Index ist die Nummer der Farbe, die animiert werden soll. Die *Verzögerung* wird in Einheiten von Fünfzigstelsekunden angegeben.

Die *Farbe* wird im RGB-Standardformat gespeichert (weitere Einzelheiten dazu finden Sie unter COLOUR). FLASH funktioniert folgendermaßen: Der Reihe nach wird jede neue Farbe auf der Liste für die durch *Verzögerung* spezifizierte Zeitspanne in den Index geladen. Wenn das Ende dieser Liste erreicht ist, wird die gesamte Farbsequenz von vorn wiederholt. Beachten Sie bitte, daß Sie mit einer FLASH-Anweisung höchstens 16 Farbänderungen durchführen können. Hier ist ein kurzes Beispiel:

```
Flash 1,"(007,10)(000,10)"
```

In diesem Beispiel wird die Farbe 1 abwechselnd alle 10/50 (1/5tel) Sekunden in schwarz bzw blau dargestellt. Und jetzt wollen wir etwas Komplexeres probieren:

```
Flash 0,"(111,2)(333,2)(555,2)(777,2)(555,4)(333,4)"
```

Wenn Ihnen das Kopfschmerzen bereitet, dann sind Sie sicher sehr erleichtert darüber, daß Sie mit der folgenden Anweisung das Blinken abstellen können:

FLASH OFF

Beachten Sie bitte auch, daß der Farbe Nummer 3 beim Starten des Systems automatisch eine Blink-Sequenz für den Cursor zugewiesen wird. Bevor Sie Bilder von der Diskette laden, schalten Sie sie vielleicht besser aus.

SHIFT UP *(Farbrotation)*

SHIFT UP Verzögerung,erste,letzte,Flag

Mit dem Befehl SHIFT UP werden die Werte in den Farbregistern von erste bis *letzte* im Rotationsprinzip verändert. Dabei wird die erste Farbe auf der Liste in die zweite, die zweite in dritte usw. kopiert, bis die letzte Farbe der Serie erreicht ist.

Jeder AMOS-Schirm kann über seine eigene Reihe von Farbanimationen verfügen. Mit Farbshifts können Sie so irre Hyperraumsequenzen erzeugen wie zum Beispiel in den Spielen Captain Blood und Elite. Da diese Animimationen unter Verwendung von Interrupts hervorgerufen werden, können Sie ausgeführt werden während Ihr Programm abläuft, ohne es im geringsten zu beeinflußen.

Die *Verzögerung* ist die Zeitspanne zwischen den einzelnen Rotationsphasen, sie wird in Fünfzigstelsekunden gemessen.

Das *Flag* bestimmt die Art der Rotation. Wenn Sie hier den Wert eins eingeben, wird der letzte Farbindex auf der Liste in den ersten kopiert und der erste wiederum in den

letzten. So kommt es zu einer ständigen Rotation der Farben auf dem Schirm. Wenn Flag den Wert Null hat, wird der Inhalt des ersten und des letzten Indexes entfernt und der Bereich zwischen ihnen allmählich durch eine Kopie der ersten Farbe auf der Liste ersetzt. Dazu ein Beispiel:

Shift Up 100,1,15,1

Shift Up 10,1,15,0

Diese Farbveränderung kann jederzeit mit dem Befehl SHIFT OFF wieder abgestellt werden.

SHIFT DOWN *(Eine Reihe von Farben wird nach unten rotiert)*

SHIFT DOWN Verzögerung,erste,letzte,Flag

SHIFT DOWN ist dem obengenannten Befehl SHIFT UP sehr ähnlich. Der Unterschied zwischen ihnen besteht darin, daß hier die Farben in entgegengesetzter Richtung rotiert werden. Also wird durch diese Anweisung die zweite Farbe in die erste, die dritte in die zweite usw. kopiert.

Erste und *letzte* dienen zur Eingabe der Farbindizes, die rotiert werden sollen. *Verzögerung* bestimmt eine Zeitspanne zwischen jeder Farbveränderung in Einheiten von Fünfzigstelsekunden.

Flag gibt wieder die Art der Rotation an. Der Wert eins führt zu einem ununterbrochenen Farbzyklus, der Wert Null verändert die Farben ohne den ursprünglichen Inhalt der Indizes *erste* und *letzte* zu speichern. Nach einem kompletten Zyklus enthalten alle Farben zwischen *erste* und *letzte* eine Kopie der in *letzte* enthaltenen Farbe. Siehe dazu auch die Befehle FLASH, PALETTE und COLOUR.

SHIFT OFF *(Stoppt alle Farbzyklen auf dem aktuellen Schirm)*

SHIFT OFF

SHIFT OFF beendet die durch die Anweisungen SHIFT UP oder SHIFT DOWN hervorgerufene Farbrotation unverzüglich.

SET RAINBOW *(Definiert einen Regenbogeneffekt)*

SET RAINBOW n,Farbe,Länge,r$,g$,b$

Mit SET RAINBOW können Sie einen interessanten Regenbogeneffekt definieren, den Sie dann danach mit dem RAINBOW-Befehl erzeugen können. Das Prinzip ist dabei die Veränderung einer Farbschattierung aufgrund einer Reihe von einfachen Regeln.

n ist die Nummer Ihres Regenbogens. Die zulässigen Werte für diesen Parameter liegen im Bereich von 0 bis 3. Farbe ist ein Farbindex, der durch diese Anweisung verändert wird. Dieser Farbe kann für jede horizontale Schirmlinie (oder Abtastlinie) ein anderer Wert zugewiesen werden. Beachten Sie aber bitte, daß nur die Farben von 0 bis 15 auf diese Art manipuliert werden können.

Länge bestimmt den Umfang der Tabelle, in der Ihre Farben gespeichert werden. In dieser Tabelle gibt es pro Farbwert auf dem Schirm jeweils nur einen Eintrag. Die Größe dieser Tabelle kann zwischen 16 und 65500 liegen. Wenn Sie für Länge einen geringeren Wert als die tatsächliche Höhe Ihres Regenbogens eingeben, so wird das Farbmuster mehrmals auf dem Schirm wiederholt.

Die Steuerketten *r$,g$,b$* ändern nacheinander die Intensität der roten, blauen und grünen Komponente Ihrer Endfarbe. Diese Werte werden in eine spezielle Farbtabelle geladen. Jede Farbe in dieser Tabelle bestimmt das Erscheinungsbild einer horizontalen Abtastlinie auf dem Schirm.

Am Anfang des Regenbogens werden alle Komponenten Ihrer Farbe erst einmal mit dem Wert Null geladen. Dann wird der Wert aufgrund der in der Farbtabelle enthaltenen Informationen verändert.

Jede der Steuerketten kann falls erforderlich auch weggelassen werden, aber Sie müssen trotzdem die Anführungszeichen und Kommas an den entsprechenden Stellen beibehalten.

Jede Kette kann eine ganze Reihe von Befehlen umfassen. Sie werden dann in einem ununterbrochenen Zyklus eingesetzt, um so das endgültige Regenbogenmuster zu erzeugen. Das Format hierzu lautet folgendermaßen:

(n,Schritt,Zähler)

n gibt die Anzahl der Zeilen an, die einem bestimmten Farbwert im Regenbogen zugeordnet wird. Wenn Sie diesen Wert erhöhen, verändert sich die Höhe jeder einzelnen Regenbogenlinie.

schritt enthält eine Zahl, die zu der Komponente addiert wird. Mit dieser Zahl wird die Farbe der nächsten Linie auf dem Schirm erzeugt. Ein positiver Wert verstärkt die Intensität der Farbkomponente und ein negativer schwächt sie ab.

Wenn eine Komponente den Höchstwert von 15 überschreitet, wird ein neuer Wert nach folgender Formel berechnet:

Neue Komponente=Alte Komponente Mod 15

zähler gibt an, wie oft die gegenwärtige Operation wiederholt werden soll. Am besten können Sie diesen Befehl wahrscheinlich anhand eines Beispiels verstehen. Geben Sie dazu bitte folgende Zeilen ein:

```
Set Rainbow 0,1,64,"(8,2,8)","",""
Rainbow 0,56,1,255 : Rem Zeigt Regenbogen an
Wait key
```

So entsteht ein neuer Regenbogen mit der Nummer Null unter Verwendung des Farbindexes eins. Um ihn nun auch auf dem Schirm anzuzeigen, müssen Sie ihn mit dem RAINBOW-Befehl (siehe unten) aufrufen.

Für den Regenbogeneffekt wird Ihre Farbe erst mit dem Wert Null geladen. Nach jeweils vier Abtastzeilen wird die rote Komponente dann automatisch um zwei erhöht. So wird sich der Inhalt der Farbe Nummer Null langsam von $000 bis $E00 ändern. Wenn die Komponente den Höchstwert von 15 übersteigt, wird der Rest berechnet und die Farbe zum Ausgangspunkt (Null) zurückgebracht. Dieses Muster wird nun nach unten auf dem Schirm wiederholt.

Durch Definieren von jeweils unterschiedlichen Mustern für die rote, blaue und grüne Komponente Ihrer Farbe können Sie leicht ganz erstaunliche Muster auf dem Schirm hervorrufen. Da jeder Regenbogen nur mit einem einzigen Farbindex arbeitet, können Sie den gleichen Effekt mit nur zwei Farbschirmen erzeugen. Das eignet sich zum Beispiel besonders gut für den Hintergrund von Spielhallen-Spielen, da auf diese Art nur wenig Speicher verbraucht wird. Hier ist ein Beispiel:

```
Screen Open 0,320,256,2,Lowres
Set Rainbow 0,1,128,"(8,1,8)","(8,1,8)",""
Rainbow 0,1,30,128
Colour 1,0 : Curs Off : Cls 1 : Flash Off
Locate 0,2 : Centre "Amos Basic" : Wait Key.
```

Ein weiteres Beispiel für die phantastischen Effekte, die Sie mit dieser Anweisung erzeugen können, finden Sie in **BEISPIEL 10.13**.

Sie können die Regenbogen auch mit einem leistungsstarken Interrupt-System animieren. In dem Kapitel über AMAL finden Sie dazu weitere Einzelheiten.

RAINBOW *(Erzeugt einen Regenbogeneffekt)*

RAINBOW n,Basis,y,h

Mit dem Befehl RAINBOW können Sie den Regenbogen mit der Nummer *n* auf dem Bildschirm anzeigen. Wenn Sie AUTOVIEW auf OFF gestellt haben, müssen Sie zuerst den Befehl VIEW aufrufen, damit der Regenbogen erscheint.

Basis ist ein Offset-Wert für die erste Farbe in der Tabelle, die Sie mit dem Befehl SET RAINBOW erstellt haben. Wenn Sie diesen Wert verändern, wird der Regenbogen über den Schirm bewegt.

y gibt die vertikale Position des Regenbogens in Hardware-Koordinaten an. Der Mindestwert für diese Koordinate beträgt 40. Wenn Sie versuchen, eine Koordinate einzusetzen, die unter diesem Wert liegt, wird der Regenbogen von Zeile 40 an angezeigt.

h stellt die Höhe Ihres Regensbogens in Abtastlinien dar.

Regenbogen sind mit dem AMOS-System, einschließlich der BOBs und Sprites, völlig kompatibel. Versuchen Sie aber nicht, eine Farbe, die gerade mit den Anweisungen FLASH oder SHIFT verändert wird, in einem Regenbogen einzusetzen,

denn das hätte völlig unberechenbare Auswirkungen auf den Schirm.

Beachten Sie bitte auch, daß auf einer bestimmten Abtastlinie nur ein einziger Regenbogeneffekt angezeigt werden kann, auch wenn auf dem Schirm andere Farben verwendet werden.

Normalerweise wird der Regenbogen mit der höchsten Schirm-Position zuerst angezeigt. Wenn aber mehrere Regenbögen an derselben Abtastlinie beginnen, wird der Regenbogen mit der niedrigsten Identifikationsnummer zuerst gezeichnet.

=RAIN *(Verändert die Farbe einer bestimmten Regenbogenlinie)*

RAIN(n,Linie)=c
c=RAIN(n,Linie)

Dies ist der stärkste aller Befehle zur Erzeugung eines Regenbogens, da Sie hier die Farbe einer bestimmten Regenbogenlinie beliebig verändern können.

n ist die Nummer des Regenbogens, den Sie bearbeiten möchten. line stellt die Abtastlinie dar, die verändert werden soll. Beispiel:

```
Curs Off:Centre "Ein AMOS Regenbogen!"
Set Rainbow 1,1,4097,"","","": Rem Erstellt einen Regenbogen mit Füllwerten
For Y=0 To 4095
   Rain(1,Y)=Y : Rem Lädt Regenbogen mit einem Farbwert aus 0-4095
Next Y
For C=0 To 4095-255
   Rainbow 1,C,40,255 : Rem Zeigt 255 Linien langen Regenbogen, angefangen bei 40
Next C
Wait Key
```

Hier wird die gesamte Palette durch die Farbe mit der Nummer eins gescrollt.

RAINBOW DEL *(Entfernt einen Regenbogen)*

RAINBOW DEL [Nummer]

Nummer gibt die *Nummer* des Regenbogens an, den Sie löschen möchten. Lassen Sie Nummer weg, so werden alle definierten Regenbogen vom Bildschirm entfernt und gelöscht.

ZOOM *(Vergrößert einen Ausschnitt des Bildschirms)*

ZOOM Quelle,x1,y1,x2,y2 TO Ziel, x3,y3,x4,y4

ZOOM ist eine einfache Anweisung, mit der Sie die Größe eines beliebigen rechteckigen Bildschirmausschnitts verändern können.

Quelle ist die Nummer des Schirms, aus dem Ihr Bild entnommen wird. Sie können das Bild auch mit der LOGIC-Funktion aus dem entsprechenden logischen Schirm entnehmen. Der rechteckige Bereich, auf den die Anweisung wirkt, wird durch die Koordinaten *x1,y1,x2,y2* definiert. *x1,y1* gibt die Position der linken oberen Ecke dieses Bereichs, und *x2,y2* die Koordinaten der diagonal gegenüberliegenden Ecke an. Ziel stellt den Zielschirm für Ihr Bild dar. Wie bei der Quelle kann auch hier wieder entweder eine Schirmnummer oder mit LOGIC ein logischer Schirm angegeben werden.

Die Abmessungen dieses Schirms werden durch die Koordinaten *x3,y3* und *x4,y4* definiert. Dies sind die Dimensionen eines Rechtecks, in das der betreffende Bildschirmausschnitt eingepaßt wird.

Welche Wirkung diese Anweisung nun tatsächlich hat, hängt von der jeweiligen Größe des Quell- bzw Zielrechtecks ab. Die Größe des Quellbildes wird automatisch den Abmessungen des Zielrechtecks angepaßt. So können Sie mit einer Anweisung Ihre Bilder entweder vergrößern oder verkleinern. Dazu folgendes Beispiel:

```
F$=Fsel$("*.*","","Load screen") : If F$="" then Direct
Load Iff F$,o : Screen Open 1,320,256, Screen Colours,Lowres
Flash off : Get Palette(0)
Screen Display 1,,,,256 : View: Limit Mouse
Repeat
   Zoom 0,070,320,175 To 1,0,0,X Screen (X Mouse)+1,Y Screen (Y Mouse)+1
Until Mouse Key
```

Ein weitere Demonstration dieses Befehls finden Sie in **BEISPIEL 10.14**.

Das Verändern der Copperliste

Durch den Coprozessor (Copper) des Amiga können Sie das Erscheinungsbild jeder einzelnen Zeile auf Ihrem Bildschirm steuern. Dieser Copper ist ein separater Prozessor, der über seinen eigenen internen Speicher und einen einzigartigen Satz von Anweisungen verfügt. Durch Programmieren des Copper kann man eine unwahrscheinliche Vielfalt von verschiedenen Schirmeffekten hervorrufen. Normalerweise wird der Copper vom AMOS-System automatisch verwaltet. Alle verfügbaren Coppereffekte können direkt in AMOS-Basic durchgeführt werden, und es besteht überhaupt keine Notwendigkeit, sich in kompliziertes Programmieren auf der Maschinenebene zu versteigen. In der Praxis sind diese Anweisungen für die Mehrzahl der Anwendungen mehr als ausreichend.

Trotzdem man kann natürlich nicht an alles denken. Die Programmierexperten unter Ihnen möchten vielleicht auf den Copper direkt zugreifen und eigene, ganz spezielle Bildschirm-Modi erzeugen.

Aber hier ist Vorsicht geboten: Die Copperliste ist berüchtigt; sie ist schwer zu programmieren, und wenn Sie nicht ganz genau wissen, was Sie tun, bringen Sie Ihren Amiga mit ziemlicher Sicherheit zum Absturz. Bevor Sie sich also zum ersten Mal auf Copper-Experimente einlassen, sollten Sie eines der vielen verfügbaren Nachschlagewerke zu diesem Thema lesen. Im Amiga System Programmers Guide von Abacus können Sie zum Beispiel eine sehr gute Erklärung finden.

COPPER OFF *(Standard-Copperliste abschalten)*

COPPER OFF

Mir dieser Anweisung wird die aktuelle AMOS Copperliste eingefroren und die Bildschirmanzeige völlig abgeschaltet. Sie können jetzt über eine Reihe von COP MOVE und COP WAIT Anweisungen Ihr eigenes Display erzeugen.

Durch den Default sind alle vom Anwender erstellten Copperlisten auf höchstens 12 KByte begrenzt. Jede Copper-Anweisung braucht durchschnittlich zwei Bytes. Sie haben also Platz für ungefähr 6000 Anweisungen. Mit einer speziellen Option aus der CONFIG-Utility kann dieser Platz aber noch vergrößert werden.

Beachten Sie bitte, daß alle Copper-Anweisungen in eine separate logische Liste geschrieben werden, die nicht auf dem Schirm angezeigt wird. Auf diese Art kann Ihr Programm das Display nicht zerstören während die Copperliste erstellt wird. Um Ihren neuen Schirm zu aktivieren, müssen Sie die physische und die logische Liste durch den Befehl COP SWAP vertauschen.

Es ist also ganz wichtig, daß Sie Ihre Copperlisten streng der Reihe nach generieren. Sie beginnen links oben auf dem Schirm und gehen dann nach rechts unten. In **BEISPIEL 10.15** im Manual-Unterverzeichnis finden Sie ein Beispiel dazu.

COPPER ON *(Copperliste neu starten)*

COPPER ON

COPPER ON startet die AMOS Copperlisten-Berechnungen wieder und zeigt die aktuellen AMOS-Schirme an. Vorausgesetzt, Sie haben seit dem Befehl COPPER OFF nichts gezeichnet, kehrt der Schirm genau in seinen Originalzustand zurück.

COP MOVE *(Schreibt eine MOVE-Anweisung in die logische Copperliste)*

COP MOVE Adr, Wert

Generiert in der logischen Copperliste eine MOVE-Anweisung.

Adr ist dabei die Adresse eines 16-Bitregisters, das geändert werden soll. Dieses Register muß in der normalen Copper-Datenzone ($7F-$1bE) liegen. Wert ist ein ganzzahliger Wert von Wortgröße, der in das bezeichnete Register geladen werden soll.

COP MOVEL *(Schreibt eine lange MOVE-Anweisung in die Copperliste)*

COP MOVEL Adr,Wert

Diese Anweisung stimmt genau mit dem normalen COP MOVE Befehl überein, außer

daß sich *addr* jetzt auf ein 32-Bitregister bezieht. value enthält einen langen (long word), ganzzahligen Wert.

COP WAIT *(Copper WAIT-Anweisung)*

COP WAIT x,y[, x Maske, y Maske]

Mit COP WAIT schreiben Sie eine WAIT-Anweisung in Ihre Copperliste. Nun wartet der Copper, bis die Hardware-Koordinaten x,y erreicht sind und gibt die Steuerung dann wieder an den Hauptprozessor ab.

Beachten Sie bitte, daß die Zeile 255 immer automatisch von AMOS verwaltet wird. Um diese Zeile brauchen Sie sich also überhaupt keine Gedanken zu machen.

x Maske und *y Maske* sind Bitmaps, die es Ihnen ermöglichen, zu warten, bis eine bestimmte Bitkombination in den Bildschirm-Koordinaten eingestellt ist. Als Default wird beiden Masken automatisch $1FF zugewiesen.

COP RESET *(Zurücksetzen des Copperlisten-Zeigers)*

COP RESET

Mit COP RESET wird die Adresse, die die nächste Copper-Anweisung verwendet, an den Anfang der Copperliste zurückgesetzt.

=COP LOGIC *(Adresse der Copperliste)*

Adr=COP LOGIC

Die Funktion COP LOGIC gibt die absolute Speicheradresse der logischen Copperliste an. So können Sie Ihre Copper-Anweisungen direkt mit POKE in den Puffer bringen, möglicherweise auch unter Verwendung der Assembly-Sprache.

Tips und Tricks

- Bevor Sie einen Schirm mit einer vom Anwender definierten Copperliste erzeugen, müssen Sie den entsprechenden Bitmaps erst etwas Speicher zuweisen. Sie können zu diesem Zweck zwar mit RESERVE arbeiten, aber es ist viel leichter, stattdessen mit SCREEN OPEN einen Füllschirm zu eröffnen. Mit der LOGBASE-Funktion können die Copper-Register dann mit den Adressen der erforderlichen Bitmaps geladen werden.

  Sie können jetzt über alle AMOS Zeichenoptionen auf den Schirm zugreifen. Um einen angemessenen Speicherplatz zu reservieren, stellen Sie die **maximale** Anzahl der Farben für den Einsatz auf dem neuen Schirm ein. Das ist zwar etwas verschwenderisch, erleichtert die Dinge aber ungemein.

- Es ist völlig akzeptabel, vom Anwender definierte Schirme mit AMOS BOBs zu kombinieren. Wenn Sie jedoch Double Buffering einsetzen, müssen Sie für den logischen und den physischen Schirm separate Copperlisten definieren. Dazu gehen Sie folgendermaßen vor:

  1 Definieren Sie Ihre Copperliste für den ersten Schirm.
  2 Vertauschen Sie die logische und physische Copperliste mit COP SWAP.
  3 Vertauschen Sie den logischen und den physischen Schirm mit SCREEN SWAP.
  4 Definieren Sie Ihre Copperliste für den zweiten Schirm.

Auf diese Art stellen Sie sicher, daß Ihre BOBs auf den neuen Schirmen korrekt aktualisiert werden. Sie können alle normalen AMOS-Befehle - einschließlich AMAL - einsetzen.



11: Hardware-Sprites

Eine der größten Attraktionen des Commodore Amiga besteht darin, daß Sie auf ihm Spiele von solch hoher Qualität entwickeln können, die selbst mit kommerziellen Spielhallen-Spielen vergleichbar sind. Dies wird von tollen Programmen wie Battle Squadron und Eliminator zur Genüge bewiesen.

Aber jetzt brauchen Sie nur noch die Hand auszustrecken, und schon sind diese phantastischen Eigenschaften greifbar nahe. Mit AMOS-Basic haben Sie vollkommene Kontrolle über die Hardware- und Software-Sprites des Amiga. Die Sprites können mit der eingebauten AMAL-Animationssprache spielend leicht bewegt werden. Sie können Ihre eigenen, begeisternden Spielhallen-Spiele schreiben, ohne irgendwelche Zauberkunststücke mit dem Maschinencode vollbringen zu müssen.

Hardware-Sprites sind separate Bilder, die automatisch auf dem Amiga-Bildschirm überlagert werden können. Das klassische Beispiel für ein Hardware-Sprite ist der Mauszeiger. Er ist vom Bildschirm des Amiga völlig unabhängig und funktioniert in allen Grafik-Modi gleich gut.

Da Sprites keine Auswirkungen auf den Bildschirmhintergrund haben, eignen sie sich besonders gut für die beweglichen Objekte, die man in einem Spielhallen-Spiel braucht. Sie sind nicht nur wahnsinnig schnell, sondern verbrauchen auch äußerst wenig Speicher. Wenn Sie also ein Spielhallen-Spiel schreiben, sollten Hardware- Sprites ganz oben auf Ihrer Liste stehen.

Jedes Sprite ist 16 Pixel breit und bis zu 255 Pixel hoch. Die Amiga Hardware unterstützt maximal acht dreifarbige Sprites oder vier fünfzehnfarbige Sprites. Die Farbe mit der Nummer Null ist transparent.

Auf den ersten Blick sind Sie vielleicht von diesen Eigenschaften nicht gerade überwältigt. Aber es gibt eine Reihe von nützlichen Tricks mit denen man sowohl Anzahl wie auch Größe der Sprites ganz irrsinnig steigern kann.

Eine Lösung besteht darin, jedes Hardware-Sprite in einige horizontale Segmente aufzuteilen. Diese Segmente können dann unabhängig voneinander positioniert werden, und so können Sie Dutzende von Sprites gleichzeitig auf dem Bildschirm darstellen. Auf ähnliche Art können Sie die vorgegebene Breite ausdehnen, indem Sie ein Objekt aus verschiedenen Einzel-Sprites zusammensetzen. Mit dieser Technik können Sie Objekte erzeugen, die bis zu 128 Pixel breit sind.

Bis vor kurzem konnte man diese Techniken nur nutzen, wenn man sich in die "Niederungen" der geheimnisvollen 68000-Assemblersprache begab. Da freuen Sie sich jetzt bestimmt, daß AMOS-Basic den ganzen Prozeß völlig automatisch durchführt! Nachdem Sie Ihre Sprites mit dem AMOS Sprite-Editor erstellt haben, können Sie sie mit einer einzigen Basic-Anweisung ganz leicht manipulieren.

Die Sprite-Befehle

Vergessen Sie nicht, daß Sie eine Sprite-Bank in den Speicher laden müssen, um die Befehle in diesem Kapitel auszuprobieren. Wir würden Ihnen raten, hier die Datei SPRITES.ABK von der AMOS-Datendiskette zu laden.

SPRITE *(Zeigt ein Hardware-Sprite auf dem Bildschirm an)*

SPRITE n,x,y,i

Der SPRITE-Befehl zeigt an den Koordinaten *x,y* ein Hardware-Sprite mit der Bildnummer *i* an.

n ist die Identifikationsnummer des Sprites und kann zwischen 0 und 63 liegen. Jedes Sprite kann mit einem eigenen Bild aus der Sprite-Bank verbunden werden, auf diese Art kann ein Bild für mehrere Sprites verwendet werden.

x und *y* geben die Position des Sprites in speziellen Hardware-Koordinaten an. Alle Abmessungen werden dem Hot Spot Ihrer Bilder entnommen. Er dient sozusagen als "Griff" des Sprites und stellt einen Referenzpunkt für die Koordinaten dar. Normalerweise befindet sich der *Hot Spot* in der linken oberen Ecke eines Bildes. Er kann jedoch in Ihrem Programm mit dem HOT SPOT Befehl auch verändert werden.

Hardware-Koordinaten sind unabhängig vom jeweiligen Schirm-Modus und beginnen bei (-129,-45) auf dem Default-Schirm. AMOS bietet Ihnen mehrere eingebaute Funktionen für die Umrechnung von Hardware- und Bildschirm- Koordinaten, wobei man mit letzteren leichter arbeiten kann. Nähere Einzelheiten darüber finden Sie bei den Funktionen X HARD, Y HARD, Y SCREEN und Y SCREEN.

I ist die Nummer eines bestimmten, in der Sprite-Bank gespeicherten Bildes. Diese Bank kann mit dem AMOS Sprite-Editor erzeugt werden und wird automatisch zusammen mit Ihrem Basic-Programm gespeichert. Sie kann mit der LOAD- Anweisung auch direkt geladen werden. Darüber hinaus können Sie mit dem Befehl GET SPRITE ein Bild sofort aus dem aktuellen Bildschirm holen.

Jeder der Parameter *x,y* und *i* kann wahlweise auch weggelassen werden, dafür **müssen** aber die entsprechenden Kommas eingesetzt werden. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Sprite 8,200,100,1
Sprite 8,,150,1
Sprite 8,300,,
```

Wenn Sie die Sprites in Aktion sehen möchten, laden Sie doch **BEISPIEL 11.1** aus dem Manual-Unterverzeichnis auf der AMOS Datendiskette.

Computed Sprites

Obwohl der Amiga Ihnen nur acht Sprites bietet, können Sie doch bis zu 64 verschiedene Objekte gleichzeitig auf dem Bildschirm darstellen. Diese Objekte nennt man *"computed", also berechnete Sprites*. Sie werden gänzlich von AMOS- Basic verwaltet. Berechnete Sprites kann man zuweisen, indem man den SPRITE- Befehl mit einer Zahl über sieben aufruft. Zum Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Sprite 8,200,100,1
```

Die Größe eines berechneten Sprites wird direkt aus den Bilddaten abgeleitet. Es kann 16 bis 128 Pixel breit, und 1 bis 255 Pixel hoch sein.

Bevor Sie diese Sprites voll ausnützen können, müssen Sie das zugrundeliegende Prinzip ein bißchen verstehen. Jedes Hardware-Sprite besteht aus einem dünnen Streifen, der 16 Pixel breit und 256 Pixel tief ist. Je nach Anzahl der Farben können Sie entweder acht oder vier dieser Streifen gleichzeitig auf dem Bildschirm haben.

DIE HARDWARE-SPRITES

S	S	S	S	S	S	S	S
P	P	P	P	P	P	P	P
R	R	R	R	R	R	R	R
I	I	I	I	I	I	I	I
T	T	T	T	T	T	T	T
E	E	E	E	E	E	E	E
1	2	3	4	5	6	7	8

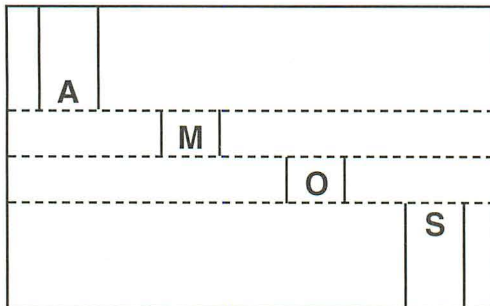
Hier sehen Sie deutlich, daß ein Großteil des Bereichs in diesen Sprites effektiv verschwendet wird. Das ist darauf zurückzuführen, daß wenige Programme Sprites benötigen, die höher als ungefähr 40 bis 64 Pixel sind. Mit einem einfachen Trick können wir uns aber diesen Platz borgen, um Dutzende von weiteren Objekten auf dem Bildschirm darzustellen. Sehen wir uns das Sprite an, das die Buchstaben A, M, O und S enthält.

EIN EINZIGES HARDWARE-SPRITE

A
M
O
S

Dieses Sprite kann man in vier horizontale Segmente aufteilen, die jeweils einen Buchstaben umfassen. Die Amiga-Hardware läßt nun zu, daß man diese Abschnitte an beliebiger Stelle auf der aktuellen Zeile positioniert, und so im Ganzen vier *berechnete* Sprites erhält. Das folgende Diagramm illustriert diesen Vorgang:

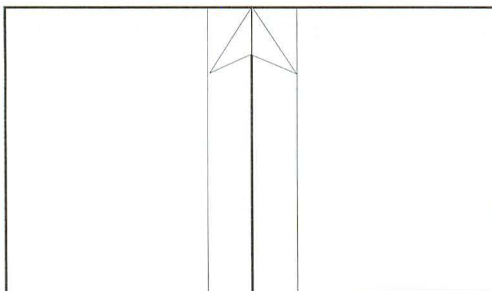
Aufteilen eines Hardware-Sprites in berechnete Sprites



Wie Sie sehen, ist ein berechnetes Sprite eigentlich nur ein kleiner Teil eines Hardware-Sprites, der an einer anderen horizontalen Position auf dem Bildschirm angezeigt wird. Beachten Sie auch die Linie zwischen den einzelnen Objekten. Sie ist eine unvermeidbare Nebenwirkung des Verschiebens, die von der Amiga- Hardware hervorgerufen wird.

Aus der Art wie berechnete Sprites erzeugt werden, ergeben sich einige Beschränkungen für Ihren Einsatz. Erstens können Sie auf einer Linie nicht mehr als acht berechnete Sprites haben. In der Praxis ergeben sich meist Komplikationen, wenn man Sprites erstellen muß, die die maximale Größe von 16 Pixel übersteigen. AMOS erzeugt diese Objekte, indem automatisch mehrere berechnete Sprites nebeneinander angeordnet werden. Das können Sie aus dem nachstehenden Diagramm gut erkennen:

Kombinieren von zwei Sprites



Hardware-Sprites

Das Maximum von acht Hardware-Sprites stellt daher eine strenge Begrenzung für die Anzahl der Objekte dar, die Sie auf einer horizontalen Linie anzeigen können. Die Gesamtbreite der Objekte darf die folgenden Abmessungen nicht überschreiten:

16*8=128 Pixel für dreifarbige Sprites

16*4=64 Pixel für fünfzehnfarbige Sprites

Wenn Sie versuchen, diese Beschränkung zu ignorieren, erhalten Sie keine Fehlermeldung. Allerdings wird Ihr berechnetes Sprite auch nicht auf dem Bildschirm angezeigt. Deshalb müssen die oben genannten Begrenzungen stets unbedingt einhalten. Dazu können Sie folgendermaßen vorgehen:

Addieren Sie die Breite aller Ihrer berechneten Sprites, dabei nehmen Sie die Abmessungen aller fünfzehnfarbigen Sprites mal zwei. Wenn das Gesamtergebnis mehr als 128 beträgt, müssen Sie Ihre Sprites so auf dem Schirm anordnen, daß die gesamte Breite unter diesem Wert liegt. Wenn Sie Ihre Sprites dann mit AMAL animieren, müssen Sie besonders vorsichtig sein, da bestimmte Kombinationen erst funktionieren, wenn Sie mit der Sequenz etwas herumexperimentiert haben.

Schlimmstenfalls müssen Sie einige Ihrer größten Sprites eben durch Blitter-Objekte ersetzen. Das wird zwar die Gesamtgröße Ihres Programmes ganz beträchtlich ausdehnen, aber die Auswirkungen, die es letztendlich auf die Qualität Ihres Programms hat, sind verschwindend gering.

Diese Beschränkungen gelten natürlich nicht nur für AMOS-Basic. Alle auf dem Amiga produzierten Spiele sind davon betroffen, selbst wenn sie ausschließlich im Maschinencode geschrieben werden! Deshalb können Sie trotzdem Ihren eigenen Klon von Xenon II mit genau denselben Techniken erzeugen.

Beachten Sie bitte, daß das Hardware-Sprite mit der Nummer Null normalerweise dem Maus-Cursor zugewiesen wird. Sie können dieses Sprite einfach durch Aufrufen des HIDE-Befehls zugänglich machen. Siehe auch **BEISPIEL 11.2**.

Das Erzeugen eines eigenen Hardware-Sprites

Das einzige echte Problem mit dem berechneten Sprites ist, daß Sie nie genau wissen, welches Hardware-Sprite in einem bestimmten Objekt verwendet wird. Normalerweise verändern sich die Hardware-Sprites, die für ein Objekt eingesetzt werden, wenn dieses Objekt bewegt wird. Manchmal kann das störend sein, vor allem, wenn Sie Objekte wie zum Beispiel Raketengeschosse animieren, die in einer großen Anzahl von verschiedenen Sprite-Kombinationen sichtbar bleiben sollen.

In diesem Fall ist es nützlich, ein Hardware-Sprite direkt zuzuweisen. Dazu verwenden Sie den SPRITE-Befehl zusammen mit einer Identifikationsnummer zwischen 0 und 7. Hier ist ein Beispiel:

```
Sprite 1,100,100,2
```

Diese Anweisung lädt ein Hardware-Sprite mit der Nummer 1 und der Bildnummer 2. N entspricht jetzt der Nummer eines einzigen Hardware-Sprites und liegt im Bereich von 0

bis 7. Wenn Ihr Bild mehr als 16 Pixel breit ist, erfaßt AMOS die erforderlichen Sprites automatisch nacheinander und beginnt dabei mit dem Sprite, das Sie ausgewählt haben. Dazu folgendes Beispiel:

```
Sprite 2,100,100,2
```

Nehmen wir an, das Bild mit der Nummer 1 enthält ein dreifarbiges 32-Bit-Bild. Mit diesem Befehl werden nun die Hardware-Sprites 2 und 3 dem Bild zugewiesen. Wenn Sie jetzt versuchten, das Hardware-Sprite 3 zum Beispiel mit dem Befehl SPRITE 3,150,100,1 anzuzeigen, würde überhaupt nichts passieren, denn dieses Sprite wurde schon verwendet. Sie könnten jetzt nur noch auf die Sprites mit den Nummern 0,1,4,5,6 und 7 zugreifen, und die maximale Anzahl und Größe Ihrer berechneten Sprites wird entsprechend reduziert.

Jedes fünfzehnfarbige Sprite wird durch ein Paar dreifarbiger Sprites implementiert. Sie können jedoch auf diese Art zwei Sprites nicht beliebig kombinieren. Nur die Kombinationen 0/1, 2/3, 4/5 und 6/7 sind zulässig. Eine Auswirkung davon ist, daß Sie immer versuchen sollten, bei der Zuweisung Ihrer Hardware-Sprites gerade Zahlen zu verwenden. Sonst beginnt AMOS Ihr Sprite mit der nächsten Zweiergruppe, und das erste Sprite wird praktisch verschwendet.

Wenn Sie ein großes, fünfzehnfarbiges Sprite erzeugen möchten, sollten Sie bitte auch beachten, daß Sie so ganz schnell alle verfügbaren Sprites für ein einziges Objekt aufbrauchen.

Achtung! Wenn Sie ein Spiel schreiben und dabei den Bildschirm scrollen, kann es Probleme dabei geben, die Sprites in Verbindung mit den Befehlen SCREEN OFFSET und SCREEN DISPLAY einzusetzen. So kommt es nämlich zu einem DMA-Konflikt zwischen dem Sprite-System und den Bitmaps des Schirms. Das kann manchmal zu unerwünschten Bildschirmeffekten führen.

Dieses Problem ist eigentlich nur von Bedeutung, wenn Sie die Hardware-Sprites 6/7 einsetzen. Wenn Sie den Schirm mit SCREEN OFFSET nach links geschoben haben, ist die Zeit, die für das Aktualisieren Ihrer Sprites zur Verfügung steht, reduziert, denn der Schirm-DMA hat vor dem Sprite-System Vorrang. Leider ist dann nicht mehr genug Zeit, die Sprites 6/7 zu zeichnen, und deshalb erscheinen Sie nur zerstört dem Bildschirm.

Um dieses Problem zu lösen, erzeugen Sie die Sprites 6/7 als individuelle Hardware- Sprites und positionieren sie mit negativen Koordinaten außerhalb des Bildschirms. Dadurch verwendet AMOS-Basic sie nicht mehr in den berechneten Sprites. Vorausgesetzt, daß die Sprites 6/7 während den Scroll-Operationen nie auf dem Bildschirm angezeigt werden, geht dann alles klar.

Die Sprite-Palette

Die für ein Hardware-Sprite benötigten Farben werden in den Farbregistern 16 bis 31 gespeichert. Wenn Ihr aktueller Bildschirm-Modus nicht auf diese Register zugreift, sind die Spritefarben von Ihren Bildschirmfarben völlig unabhängig. Interessanterweise ist das auch für den 4096-farbigen HAM-Modus der Fall. Also stehen Ihnen alle Möglichkeiten offen, mit diesem System absolut umwerfende HAM-Spiele zu erzeugen.

Wenn Sie jedoch versuchen, Schirme mit 32 oder 64 Farben in Verbindung mit dreifarbigen Sprites zu verwenden, kann die Sache ziemlich problematisch werden. Das ist darauf zurückzuführen, daß die Farben, die diese Sprites einsetzen, auf folgende Art gruppiert sind:

Hardware-Sprites	Farbregister
0/1	17/18/19
2/3	21/22/23
4/5	25/26/27
6/7	29/30/31

Die Farbregister 16,20,24 und 28 stellen immer die transparente Farbe dar.

Die Schwierigkeiten entstehen nun durch die Art, wie AMOS berechnete Sprites erzeugt. Die Hardware-Sprites, aus denen diese Objekte bestehen, verändern sich im Laufe eines Spiels, und deshalb müssen Sie unbedingt sicherstellen, daß für die drei Farben, die von jedem einzelnen Sprite verwendet werden, derselbe Farbwert eingegeben wird. Andernfalls verändern sich die Farben Ihrer berechneten Sprites völlig willkürlich. Hier folgt nun eine kleine AMOS-Prozedur, die den ganzen Vorgang automatisch für Sie ausführt.

```
Procedure INIT_SPRITES
   Get Sprite Palette
   For S=0 To 3
         For C=0 To 2
               Colour S*4+C+17,Colour(c)
         Next C
   Next S
Endproc
```

Die obengenannte Beschränkung trifft natürlich nicht auf fünfzehnfarbige Sprites zu. Wenn Sie alle Möglichkeiten der Extra-Half-Bright- oder 32-Farb-Modi ausschöpfen möchten, ist es einfacher, den Einsatz von vierfarbigen Sprites völlig zu vermeiden.

GET SPRITE PALETTE *(Lädt die Spritefarben in den Bildschirm)*

GET SPRITE PALETTE [Maske]

Dieser Befehl lädt die gesamte Farbpalette, die für Ihre Sprite-Bilder verwendet wird, in den aktuellen Bildschirm. Der Parameter Maske, den Sie wahlweise eingeben können, ermöglicht es Ihnen, nur eine Auswahl der Farben aus der Sprite- Palette zu laden. Jede der 32 Farben wird von einem Bit in der Maske dargestellt, wobei die Numerierung von links nach rechts erfolgt. Das Bit ganz rechts stellt den Status der Farbe Null dar, das

danebenliegende Bit die Farbe 1 usw. Um eine Farbe zu laden, stellen Sie einfach das entsprechende Bit auf 1. Wenn Sie zum Beispiel nur die ersten vier Farben kopieren möchten, geben Sie folgendes Bitmuster ein:

Get Sprite Palette %0000000000001111

Übrigens, da BOBs auf dieselben Banken zugreifen wie die Sprites, können Sie mit diesem Befehl auch die Farben eines BOBs laden.

Das Steuern von Sprites

SET SPRITE BUFFER *(Bestimmt die Höhe der Hardware-Sprites)*

SET SPRITE BUFFER n

Hiermit wird der Arbeitsbereich definiert, in dem AMOS die Abbilder der Hardware-Sprites erzeugt. Die zulässigen Werte für *n* liegen im Bereich von 16 bis 256. Um den korrekten Wert für *n* einzugeben, sehen Sie sich die Sprites im Sprite-Editor an und finden heraus, welches davon die größte Länge aufweist. Jedes Sprite, das größer als *n* ist, wird einfach am entsprechenden Punkt abgeschnitten.

SET SPRITE BUFFER dient dazu, daß Sie auf den Speicher, der von Ihrem Spiel oder Ihrer Anwendung nicht genutzt wird, wieder zugreifen können.

Mit der folgenden Formel können Sie den Speicherplatz ermitteln, der vom Sprite-Puffer belegt wird:

Speicher = N*4*8*3 = N*96

So beträgt also der kleinste Puffer 1536 Bytes und das Maximum liegt bei 24 KBytes. Beachten Sie bitte: Durch diesen Befehl werden alle aktuellen Sprite- Zuweisungen gelöscht und der Maus-Cursor wieder auf seinen Ausgangspunkt zurückgesetzt.

SPRITE OFF *(Ein oder mehrere Sprites vom Bildschirm entfernen)*

SPRITE OFF [n]

Mit dem Befehl SPRITE OFF können Sie ein oder mehrere Sprites vom Bildschirm entfernen. Alle gegenwärtigen Sprite-Bewegungen werden abgebrochen. Um sie wieder aufzunehmen, müssen Sie Ihre Bewegungsmuster neu initialisieren.

SPRITE OFF	Entfernt alle Sprite vom Display.
SPRITE OFF n	Setzt nur das Sprite mit der Nummer n außer Kraft.

Jedesmal, wenn Sie den AMOS Basic-Editor aufrufen, werden Ihre Sprites automatisch außer Kraft gesetzt. Wenn Sie dann das nächste Mal wieder in den Direktmodus gehen, kehren die Sprites wieder an ihre ursprüngliche Position zurück.

SPRITE UPDATE *(Steuert Sprite-Bewegungen)*

SPRITE UPDATE [ON/OFF]

Der Befehl SPRITE UPDATE gibt Ihnen die totale Kontrolle über die Bewegungen Ihrer Sprites. Immer wenn Sie ein Sprite bewegen, wird seine Position normalerweise während der nächsten Vertical Blank Period (siehe WAIT VBL) aktualisiert. Wenn Sie nun aber eine ganze Reihe von Sprites mit dem SPRITE- Befehl bewegen, wird die Aktualisierung durchgeführt, bevor alle Sprites verschoben sind. Das kann zu einem merklichen Ruckeln in Ihren Bewegungen führen. In diesem Fall können Sie das automatische Aktualisierungs-System mit dem Befehl SPRITE UPDATE OFF ausschalten.

Nachdem dann Ihre Sprite erfolgreich verschoben wurden, können Sie sie mit der Anweisung SPRITE UPDATE ON ganz leicht auf ihren Platz gleiten lassen. Durch diesem Befehl werden alle Sprites, die seit der letzten Aktualisierung bewegt wurden, neu positioniert.

Siehe auch UPDATE EVERY, UPDATE und BOB UPDATE.

=X SPRITE *(Gibt die x-Koordinate eines Sprites an)*

x = X SPRITE(n)

Weist die aktuelle x-Koordinate des Sprites mit der Nummer n in Hardware- Koordinaten aus. Mit diesem Befehl können Sie schnell überprüfen, ob ein Sprite über den Rand des Amiga-Bildschirms gerutscht ist.

=Y SPRITE *(Gibt die y-Koordinate eines Sprites an)*

y = Y SPRITE(n)

Y SPRITE gibt die vertikale Position eines Sprites an. Wie üblich bezieht sich *n* auf die Nummer des Sprites. Sie kann zwischen 0 und 63 liegen. Vergessen Sie nicht, daß alle Sprite-Positionen in Hardware-Koordinaten angegeben werden. Siehe auch **BEISPIEL 11.3**.

GET SPRITE

(Lädt einen Bildschirmabschnitt in die Sprite-Bank)

GET SPRITE [s,]i,x1,y1 To x2,y2

Mit dieser Anweisung können Sie Bilder direkt vom Bildschirm holen und in Sprites verwandeln. Die Koordinaten *x1,y1* und *x2,y2* definieren einen rechteckigen Bereich, der in der Sprite-Bank gespeichert werden soll. Normalerweise werden alle Bilder auf dem aktuellen Schirm erfaßt. Sie können aber auch durch Eingeben der Schirmnummer s ein Bild aus einem bestimmten Schirm erfassen.

Beachten Sie bitte: Der Bereich, der auf diese Weise erfaßt werden kann, unterliegt keinen Beschränkungen. Vorausgesetzt Ihre Koordinaten liegen innerhalb der bestehenden Bildschirmgrenzen, geht alles völlig glatt.

I stellt die Nummer des neuen Bildes dar. Wenn kein Sprite mit dieser Nummer vorhanden ist, so wird automatisch ein neues Bild erzeugt. AMOS übernimmt auch das Reservieren der Sprite-Bank, sofern sie nicht vorher schon definiert wurde. Siehe hierzu **BEISPIEL 11.4**.

Es gibt auch eine entsprechende GET BOB Anweisung, die mit GET SPRITE in jeder Beziehung übereinstimmt. Da die Sprite-Bank sowohl BOBs wie auch Sprites enthält, verfügen die Bilder genau über das gleiche Format. Deshalb können Sie beide Anweisungen für BOBs ebenso wie für Sprites einsetzen. Versuchen Sie doch einmal, die Sprite-Anweisung im obengenannten Beispiel folgendermaßen zu verändern:

```
Bob 1,0,0,1
```

Konvertierungs-Funktionen

=X SCREEN
=Y SCREEN

(Wandelt die Hardware-Koordinaten in Bildschirm-Koordinaten um)

X=X SCREEN([n,]xcoord)
Y=Y SCREEN([n,]ycoord)

Konvertiert eine Hardware-Koordinate in eine Bildschirm-Koordinate mit Bezug auf den aktuellen Schirm. Wenn die Hardware-Koordinate außerhalb des Bildschirms liegt, so weisen beide Funktionen den relativen Offset von den Grenzflächen des Bildschirms aus. Geben Sie dazu das folgenden Beispiel aus dem Direktmodus ein:

```
Print X Screen(130)
```

Das Ergebnis beträgt -2. Das kommt dadurch zustande, weil die x-Koordinate 0 des Bildschirms der Hardware-Koordinate 128 entspricht. Daher ergibt die Umwandlung des Wertes 130 in die Bildschirm-Koordinate eine Position von zwei Pixel links vom Bildschirm.

Wenn Sie für den Parameter n einen Wert einsetzen, dann werden die Koordinaten in Bezug zum Bildschirm mit dieser Nummer ausgewiesen.

=X HARD
=Y HARD

(Wandelt Bildschirm-Koordinaten in Hardware-Koordinaten um)

X=X HARD ([n,] xcoord)

Diese Funktionen wandeln Bildschirm-Koordinaten in Hardware-Koordinaten um. Es gibt nur vier verschiedene Konvertierungs-Funktionen, und die obengenannte Syntax wandelt die xcoord aus einer Bildschirm-Koordinate mit Bezug zum aktuellen Schirm in eine Hardware-Koordinate um.

Y=Y HARD([n,] ycoord)

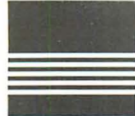
Wandelt eine y-Koordinate mit Bezug zum aktuellen Schirm in eine Hardware-Koordinate um. Wie zuvor kann *n* dabei einen bestimmten Schirm bezeichnen. Alle Koordinaten beziehen sich dann auf diesen Schirm.

=I SPRITE

Gibt das aktuelle Bild eines Sprites an)

Image=I Sprite(n)

Diese Funktion weist die aktuelle Bildnummer aus, die von dem Sprite mit der Nummer n verwendet wird. Wenn das Sprite nicht angezeigt wird, dann wird hier ein Wert von Null angegeben.



12: Blitter-Objekte

Während Hardware-Sprites zweifelsohne sehr leistungsstark sind, unterliegen sie doch einer Reihe von ärgerlichen Beschränkungen. So können zum Beispiel nur höchstens acht Sprite auf einer horizontalen Linie dargestellt werden, und jedes Sprite ist auf fünfzehn Farben beschränkt.

Die Lösung besteht darin, den berüchtigten Blitter-Chip des Amiga auszunutzen. So können Bilder mit nahezu einer Million Pixel pro Sekunde auf den Bildschirm kopiert werden. Mit Hilfe des Blitters können Sie die sogenannten BOBs (oder Blitter-Objekte) erzeugen.

BOBs, ähnlich wie Sprites, können völlig unabhängig vom Bildschirm bewegt werden, ohne daß existierende Grafiken zerstört werden. Aber anders als Sprites werden BOBs als Teil des aktuellen Schirms gespeichert, und Sie können sie in jedem beliebigen Grafikmodus erzeugen. So können Sie Blitter-Objekte mit bis zu 64 Farben generieren. Darüberhinaus ist die Anzahl der BOBs, die Sie anzeigen können, nur durch den verfügbaren Speicher begrenzt.

Natürlich hat auch leider diese Leistung ihren Preis. In der Praxis sind BOBs etwas langsamer als Sprites und sie fressen wesentlich mehr Speicher. Daher sind Sie vielleicht zwischen der Geschwindigkeit der Sprites und der Flexibilität der BOBs hin- und hergerissen. Aber glücklicherweise können Sie ja BOBs und Sprites nebeneinander im selben Programm einsetzen. Diese Lösung wird bei den meisten der handelsüblichen Spiele verwirklicht. BOBs werden für größere Objekte wie zum Beispiel Raumschiffe eingesetzt, während Hardware-Sprites oft für kleine, schnelle Objekte wie Raketengeschoße verwendet werden.

BOB *(Zeichnet ein Blitter-Objekt auf den aktuellen Schirm)*

BOB n,x,y,i

Der BOB-Befehl erzeugt das BOB *n* an den Koordinaten *x,y* mit der Bildnummer *i*.

n stellt dabei die Identifikationsnummer des BOBs dar. Die zulässigen Werte liegen normalerweise im Bereich von 0 bis 63, aber die Anzahl der BOBs kann über eine Option aus dem AMOS Konfigurations-Programm erhöht werden. Sie diese Grenze beliebig ausdehnen, vorausgesetzt es steht Ihnen genug Speicher zur Verfügung.

x und *y* geben die Position des BOBs in normalen Bildschirmkoordinaten an. Diese Koordinaten unterscheiden sich von den Hardware-Koordinaten, die vom entsprechenden SPRITE-Befehl verwendet werden. Genau wie ein Sprite wird auch jedes BOB durch einen *Hot Spot* gesteuert. Mit dem Befehl HOT SPOT kann dieser jederzeit verändert werden.

i bezieht sich auf ein Bild, das dem BOB von der Sprite-Bank zugewiesen wird. Das Format dieses Bildes ist identisch mit dem, das für die Sprites eingesetzt wird. Sie können also dasselbe Bild für Sprites und BOBs verwenden.

Nach dem Erstellen eines BOBs können Sie seine Position oder sein Erscheinungsbild einfach durch das Weglassen eines oder mehrerer Parameter aus dieser Anweisung verändern. Jeder der Werte *x,y* oder *i* kann hier wegfallen, und die

weggelassenen Parameter behalten dabei ihren ursprünglichen Wert. Das ist ganz besonders nützlich, wenn Sie Ihr BOB mit AMAL animieren. Sie können dann nämlich Ihre Objekte beliebig verschieben, ohne die bestehende Animations- Sequenz zu beeinflußen. Sie müssen die Kommas jedoch immer in ihrer ursprünglichen Reihenfolge einsetzen, sonst erhalten Sie einen Syntax-Fehler. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Flash Off : Get Sprite Palette
Channel 1 To Bob 1
Bob 1,0,100,1
Amal 1,"Anim 0,(1,4)(2,4)(3,4)(4,4)"
Amal On
For X=1 To 320
  Bob 1,X,,
  Wait Vbl
Next X
```

Immer, wenn Sie ein BOB bewegen, wird der darunterliegende Bereich in seiner ursprünglichen Position ersetzt. Das hat genau dieselbe Wirkung wie der entsprechende Sprite-Befehl. Anders als bei STOS auf dem ST wird jedes Objekt einem eigenen Speicherbereich zugeordnet. Auf diese Art wird der von den BOBs verwendete Speicher reduziert, und die Gesamtleistung drastisch verbessert. Durch den Blitter kann man natürlich die STOS Sprites und AMOS BOBs eigentlich nicht miteinander vergleichen.

Der BOB-Befehl funktioniert bei einer kleinen Anzahl von BOBs zwar sehr gut, aber wenn Sie versuchen, mehr als drei oder vier Objekte auf dem Bildschirm gleichzeitig einzusetzen, tritt ein äußerst störendes Flackern auf. Das ist darauf zurückzuführen, daß die BOBs zur selben Zeit wie der Bildschirm aktualisiert werden. Daher können Sie die BOBs sehen, während sie gezeichnet werden, und das führt dann zu dem unangenehmen Flackern.

Sie können die Qualität Ihrer Animationen zum Beispiel verbessern, indem Sie Ihre BOBs auf das untere Viertel des Bildschirms beschränken. Da BOBs überaus schnell neu gezeichnet werden, kann die Aktualisierung oft durchgeführt werden, bevor der untere Teil des Bildschirms angezeigt wird. So erhalten Sie ganz ordentliche glatte Bewegungen und verbrauchen dabei wenig Speicher. Wenn der Platz also knapp wird, dann ist das ein ganz nützlicher Trick. Siehe auch **BEISPIEL 12.1**.

Natürlich stellt das keine richtige Lösung für solch ein ernsthaftes Problem dar. Bevor Sie also entsetzt Ihr AMOS-Basic wegwerfen, hören Sie bestimmt gern, daß Sie dieses Flimmern auf ganz einfache Art abstellen können, selbst wenn Sie Dutzende von BOBs auf dem Bildschirm verwenden.

DOUBLE BUFFER *(Erzeugt einen Bildschirm-Doppelpuffer)*

DOUBLE BUFFER

Mit dem Befehl DOUBLE BUFFER erzeugen Sie eine zweite, unsichtbare Ausführung

des aktuellen Bildschirms. Alle Grafik-Operationen, einschließlich der BOB-Bewegungen, werden jetzt direkt im *logischen* Schirm ausgeführt, und haben somit überhaupt keine Auswirkung auf Ihr Fernsehbild.

Nachdem das Bild einmal neu gezeichnet ist, wird der logische Schirm angezeigt, und der ursprüngliche *physische* Bildschirm wird der neue logische Schirm. Dieser ganze Prozeß läuft ununterbrochen ab, und erzeugt eine völlig ruckelfreie Anzeige, selbst wenn Sie Hunderte von BOBs gleichzeitig auf dem Bildschirm herumschieben.

Diese ganze Prozedur wird von AMOS-Basic automatisch durchgeführt. Wenn Sie also die Anweisung einmal ausgeführt haben, können Sie sie völlig vergessen. Da die Hardware-Sprites immer auf dem aktuellen physischen Bildschirm angezeigt werden, hat dieses System überhaupt keine Auswirkungen auf die vorhandenen Sprite-Animationen.

Double-Buffering funktioniert in allen Amiga Grafik-Modi gleich gut. Sie können diese Technik sogar in Verbindung mit Dual Playfields einsetzen. Hier jedoch eine Wort der Warnung: Der Speicherplatz, den Ihre Schirme verbrauchen, wird durch das Double-Buffering verdoppelt. Wenn Sie versuchen, es für zuviele Schirme einzusetzen, dann wird Ihnen bald der Speicher knapp werden. Siehe dazu **BEISPIEL 12.2**.

In der Praxis ist Double-Buffering eine unwahrscheinlich nützliche Technik, die für die meisten Arten von Spielen äußerst gut eingesetzt werden kann. So wird sie in einer Mehrheit der handelsüblichen Spiele - einschließlich Starglider - verwendet. Und deshalb stellt sie auch einen wichtigen Bestandteil von AMOS-Basic dar. Eine detaillierte Erklärung zu diesem Prozeß finden Sie in dem Kapitel über BILDSCHIRME. Siehe außerdem die Befehle SCREEN SWAP und AUTOBACK.

SET BOB *(Stellt den Zeichenmodus für BOBs ein)*

SET BOB n, Hinter, Planes, Minterms

Der Befehl SET BOB verändert den Zeichenmodus, der zur Darstellung eines Blitter-Objektes auf dem Bildschirm verwendet wird. *n* ist dabei die Nummer des BOBs, auf das diese Anweisung wirken soll.

hinter bestimmt die Art, auf die der Hintergrund Ihres BOBs neu gezeichnet wird. Es gibt hier drei Möglichkeiten:

- Der Wert Null zeigt an, daß der Bereich unter Ihrem BOB gespeichert werden soll. Die alten Bilddaten werden automatisch ersetzt, wenn der BOB bewegt wird, und so kommt eine fließende Bewegung zustande.

- Wenn der Parameter *hinter* positiv ist, wird der ursprüngliche Hintergrund völlig ersetzt und der Bereich hinter dem BOB wird mit der Farbe *Hinter-1* ausgefüllt. Das ist ideal für das Schieben eines BOBs über einen geschlossenen Farbblock wie zum Beispiel einen blauen Himmel geeignet, da es viel schneller ist als das normale Zeichensystem.

- Schaltet das Neuzeichnen völlig aus, wenn Hinter mit einem negativen Wert wie zum Beispiel -1 geladen wird. Sie können jetzt den automatischen Aktualisierungsprozeß

mit BOB UPDATE ausschalten, und Ihre BOBs manuell mit BOB DRAW verschieben. So können Sie den Bildschirm-Hintergrund mit Ihren eigenen, maßgeschneiderten Zeichenroutinen neu erstellen.

Planes ist eine Bitmap, die AMOS mitteilt, in welchen Bildschirmplanes Ihr BOB gezeichnet wird. Wie Sie vielleicht wissen, ist der Bildschirm des Amiga in eine Reihe separater Bitplanes aufgeteilt. Jede Plane bestimmt genau ein Bit in der Endfarbe, die auf dem Bildschirm erscheint.

Die erste Plane wird von Bit eins dargestellt, die zweite von Bit zwei usw. Normalerweise wird das BOB in allen Bitplanes im aktuellen Schirm-Modus gezeichnet. Das entspricht einem Bitmuster von %111111.

Wenn Sie nun einige dieser Bits auf Null stellen, können Sie bestimmte Farben in Ihren BOBs beim Zeichen weglassen. So können Sie eine Reihe ganz faszinierender Bildschirmeffekte hervorrufen.

Minterms wählt den Blitter-Modus, mit dem Ihre BOBs auf dem Bildschirm gezeichnet werden. In dem Abschnitt zum Befehl SCREEN COPY finden Sie eine umfassende Beschreibung der verfügbaren Zeichen-Modi. *Minterm* wird normalerweise auf einer der beiden folgenden Werte gestellt:

%11100010	Wenn der BOB mit einer Maske verwendet wird
%11001010	Wenn NO MASK eingestellt wurde

Probieren Sie die verschiedenen Kombination ruhig einmal aus. Es besteht gar keine Gefahr, daß Ihr Amiga abstürzt, wenn Sie einen Fehler machen. Die folgenden Informationen werden vor allem für fortgeschrittene Amiga-Anwender nützlich sein.

Blitter-Source	**Zweck**
A	Blitter-Maske
B	Blitter-Objekt
C	Zielbildschirm

Beachten Sie bitte, daß es empfehlenswert ist, SET BOB einzusetzen, **bevor** Sie Ihre BOBs auf dem Bildschirm anzeigen. Wenn Sie das nicht tun, stürzt der Amiga zwar nicht ab, und Sie erhalten auch keine Fehlermeldung, aber Ihre Anzeige kann zerstört werden.

NO MASK *(Entfernt Blitter-Maske)*

Farbe 0 ist transparent
→ nicht "

NO MASK [n]

Als Default wird automatisch eine Blitter-Maske für jeden BOB erstellt, das Sie auf dem Bildschirm anzeigen. Diese Maske wird mit dem Bildschirmhintergrund verbunden, um die Farbe Null transparent zu machen. Sie wird auch von den verschiedenen Befehlen zur Kollisionsabfrage eingesetzt.

Mit dem Befehl NO MASK wird diese Maske nun entfernt, und das ganze Bild zwangsläufig auf dem Bildschirm gezeichnet. Alle Teile des Bildes in der Farbe Null werden jetzt direkt auf dem vorhandenen Hintergrund angezeigt.

n ist die Nummer des Bildes, dessen Maske entfernt wird. Diese Maske sollte nie gelöscht werden, wenn das Bild aktiv auf dem Bildschirm dargestellt wird, sonst wird der entsprechende BOB zerstört. Wenn Sie die Maske mit dieser Anweisung entfernen müssen, sollten Sie die betreffenden BOBs unbedingt zuerst mit dem Befehl BOB OFF deaktivieren. Hier ist ein Beispiel dafür:

```
Centre "Maustaste anklicken und Maske entfernen"
Double Buffer : Load "AMOS_DATA:Sprites/Octopus.abk" : Get Sprite palette
Do
  Bob 1,X Screen(X Mouse),Y Screen(Y Mouse),1
  If Mouse Click Then Bob Off : No Mask 1
Loop
```

Siehe auch MAKE MASK.

AUTOBACK *(Stellt das automatische Kopieren der Bildschirme ein)*

AUTOBACK n

Wenn Sie mit einem Double-Buffer-Schirm arbeiten, ist es unerläßlich, daß Sie Ihre Zeichenoperationen mit den Bewegungen Ihrer Blitter-Objekte synchronisieren. Vergessen Sie dabei nicht, daß jeder Double-Buffer-Schirm aus zwei getrennten Displays besteht. Ein Schirm zeigt das aktuelle Bild an, während der andere dazu dient, das Objekt zu erstellen. Wenn der Hintergrund eines BOBs sich verändert, während er neu gezeichnet wird, dann ist auch der Inhalt dieser Schirme anders. Das führt zu einem starken und störenden Flimmern.

Das einzigartige AMOS AUTOBACK-System bietet Ihnen die perfekte Lösung für dieses Problem. Mit seiner Hilfe können Sie Ihre Grafiken in jedem beliebigen der drei Grafikmodi generieren, je nach den jeweiligen speziellen Anforderungen Ihres Programms. Diesmal wollen wir das Pferd zur Abwechslung einmal von hinten aufzäumen und beginnen also mit der dritten Option.

AUTOBACK 2 (Automatik-Modus - Default)

In diesem Modus werden alle Zeichenoperationen automatisch mit der Aktualisierung der BOBs verbunden. So wird alles, was Sie auf dem Bildschirm zeichnen wie durch Zauberei direkt unter Ihren BOBs angezeigt. Die diesem System zugrundeliegenden Prinzipien können durch folgenden Code demonstriert werden.

```
Rem Auf ersten Schirm zeichnen
Rem Bobs entfernen
Bob Clear
```

```
Rem Grafikoperation durchführen
Plot 150,100 : Rem Das kann alles sein, was man will
Bob Draw : Rem Bobs neu zeichnen
Screen Swap : Rem Nächster Schirm
Wait Vbl
Rem Objekt auf den nächsten Schirm zeichnen
Bob Clear
Plot 150,100 : Rem Operation ein zweites Mal durchführen
Bob Draw
Screen Swap : Rem Zum ersten Schirm zurückgehen
Wait Vbl
```

Wie Sie sehen, werden alle Bildschirm-Aktualisierungen genau zweimal durchgeführt. Es gibt jeweils eine Operation für den logischen und für den physischen Bildschirm. In **BEISPIEL 12.3** erhalten Sie eine Demonstration.

Eine offensichtliche Nebenwirkung besteht darin, daß das Zeichnen Ihrer Grafiken jetzt doppelt so lang dauert. Außerdem wird das Programm jedesmal, wenn Sie etwas auf dem Bildschirm ausgeben, durch mindestens 2 Vertikal Blanks (oder 2/50stel Sekunden) angehalten. Das kann zu störenden Verzögerungen von kritischen Aktivitäten wie zum Beispiel die Kollisionsabfrage führen.

AUTOBACK 1 (Halbautomatischer Modus)

Führt jede Grafikoperation sowohl auf dem physischen wie auch dem logischen Schirm durch. Blitter-Objekte werden dabei überhaupt nicht berücksichtigt, deshalb sollten Sie dieses System nur für das Zeichnen außerhalb der aktuellen Spielfläche verwenden.

Anders als im Standard-Modus besteht keine Notwendigkeit, Ihr Programm bis zum nächsten Vertical Blank anzuhalten. Der Modus 1 eignet sich daher besonders für Schalter oder Punktstandtabellen, die während des Spiels dauernd aktualisiert werden müssen.

AUTOBACK 0 (Manueller Modus)

Bringt das AUTOBACK-System völlig zum Stillstand. Jetzt werden alle Grafiken mit Höchstgeschwindigkeit direkt auf dem logischen Schirm ausgegeben. Sie sollen diese Option einsetzen, wenn Sie ständig große Teile Ihres Hintergrundschirms während eines Spiels immer wieder neu zeichnen müssen. Mit dieser Option können Sie Ihre Routinen zur Abfrage von Kollisionen in regelmäßigen Abständen problemlos durchführen, ohne die Gesamtqualität der Animationseffekte zu zerstören. Sie das gut anhand der folgenden Programmschleife untersuchen.

```
Bob Update Off : Rem Automatisches Aktualisieren abstellen
Repeat
Screen Swap
Wait Vbl
Rem Bobs vom Schirm löschen
Bob Clear
```

```
Rem
Rem Jetzt alle Grafiken, die verändert wurden, neu zeichnen
Rem Spielroutinen (Kollisionsmeldungen etc) durchführen
Rem Bob-Bilder aktualisieren
Rem
Bob Draw
Rem Physischen und logischen Schirm austauschen
Until WIN : Rem Weitermachen bis Spiel zu Ende ist
```

Beachten Sie bitte, daß diese Prozedur **nur** funktioniert, wenn ein fließender Übergang von Schirm zu Schirm gewährleistet ist. Sie müssen die Inhalte von physischem und logischem Schirm miteinander abstimmen. In **BEISPIEL 12.4** finden Sie ein Beispiel für diese Technik.

Nehmen wir einen Moment an, Sie möchten ein BOB über einer Reihe von willkürlichen Blöcken darstellen. Dazu könnten Sie zum Beispiel die folgende Routine einsetzen:

```
Load "AMOS_DATA:Sprites/Octopus.abk" : Flash Off : Get Sprite PaletteDouble Buffer : Cls 0
Double Buffer : Cls 0 : Autoback 0 : Update Off
Bob 1,160,100,1
Do
   Bob Clear
   X=Rnd(320)+1:Y=Rnd(200)+1:W=Rnd(80)+1:H=Rnd(50)+1: I=Rnd(15)
   Ink I : Bar X,Y To X+W,Y+H
   Bob Draw
   Rem Zweiten Schirm aktualisieren
   Screen Swap : Wait Vbl
   Bob Clear
   Ink I : Bar X,Y To X+W,Y+H
   Bob Draw
Loop
```

Da aber zwischen dem physischen und dem logischen Schirm keine Beziehung besteht, springt das Display jetzt ständig von Schirm zu Schirm. Dieses Problem können Sie lösen, indem Sie das ursprünglichen AUTOBACK-System folgendermaßen nachahmen:

```
Load "AMOS_DATA:Sprites/Octopus.abk" : Flash Off : Get Sprite Palette
Double Buffer : Cls 0 : Autoback 0 : Update Off
Bob 1,160,100,1
Do
  Rem Ersten Schirm aktualisieren
  Screen Swap : Wait Vbl
  Bob Clear
  X=Rnd(320)+1:Y=Rnd(200)+1:W=Rnd(80)+1:H=Rnd(50)+1: I=Rnd(15)
  Ink I : Bar X,Y To X+W,Y+H
  Bob Draw
```

```
    Rem Zweiten Schirm aktualisieren
    Screen Swap : Wait Vbl
    Bob Clear
    Ink I : Bar X,Y To X+W,Y+H
    Bob Draw
  Loop
```

Die beiden Schirme werden jetzt mit genau derselben Information aktualisiert, und die Anzeige bleibt ruckelfrei bestehen, obwohl im Hintergrund ziemlich viel passiert.

Sie können AUTOBACK ganz ohne Bedenken an jeder beliebigen Stelle in Ihrem Programm einsetzen. Es ist also gut möglich, unterschiedliche Zeichenmethoden für verschiedene Abschnitte Ihres Bildschirms zu verwenden. Diese Optionen sind auch mit allen Grafikoperationen einschließlich der Block-, Icon- und Fensterfunktionen völlig kompatibel.

Die Steuerung der BOBs

BOB UPDATE *(Steuerung der BOB-Bewegungen)*

BOB UPDATE [ON/OFF]

Normalerweise werden alle BOBs alle 50stel Sekunden über eine eingebaute Interrupt-Routine aktualisiert. Obwohl sich dies für die meisten Programme ganz gut eignet, muß das Neuzeichnen bei einigen Anwendungen viel feiner gesteuert werden.

Mit dem Befehl BOB UPDATE OFF können Sie die Aktualisierung der BOBs und das gesamte automatische Vertauschen der Bildschirme, wenn Sie das zuvor gewählt hatten, abschalten. Sie können Ihre BOBs jetzt mit dem Befehl BOB UPDATE an dem geeignetsten Punkt in Ihrem Programm neu zeichnen. Das ist ideal, wenn Sie eine große Anzahl von Objekten animieren, da Sie so Ihre BOBs in Position bringen können, bevor sie auf den Bildschirm gezeichnet werden. Das führt zwangsläufig zu glatten Bewegungsabläufen in Ihren Spielen.

Hier ist jedoch etwas Vorsicht geboten: Die BOBs werden jetzt erst beim **nächsten** Vertical Blank aktualisiert. Beachten Sie auch, daß BOB UPDATE die BOBs stets auf dem aktuellen logischen Schirm neu zeichnet. Wenn Sie also vergessen, den Befehl SCREEN SWAP einzusetzen, dann passiert scheinbar garnichts.

BOB CLEAR *(Entfernt alle BOBs vom Bildschirm)*

BOB CLEAR

Die Anweisung BOB CLEAR entfernt alle aktiven BOBs vom Bildschirm und zeichnet die darunterliegenden Bereiche neu. In Verbindung mit BOB DRAW stellt diese Anweisung eine Alternative zum normalen BOB UPDATE Befehl dar.

BOB DRAW *(Zeichnet BOBs neu)*

BOB DRAW

Wenn die BOBs auf dem Bildschirm neu gezeichnet werden, werden die folgenden Schritte stets automatisch ausgeführt:

1 Alle aktiven BOBs werden vom LOGISCHEN Schirm entfernt und die Hintergrundbereiche ersetzt. Dieser Schritt wird durch BOB CLEAR ausgelöst.
2 Alle BOBs, die seit der letzten Aktualisierung bewegt wurden, werden aufgelistet.
3 Die Hintergrundbereiche werden unter den neuen Bildschirm-Koordinaten gespeichert.
4 Alle aktiven BOBs werden an Ihren neuen Positionen auf dem logischen Schirm neu gezeichnet.
5 Wenn die DOUBLE-BUFFER-Funktion aktiviert wurde, werden jetzt der physische und der logische Schirm miteinander vertauscht.

Der Befehl BOB DRAW führt die Schritte 2 bis 4 dieses Vorgangs direkt aus. Angenommen Sie möchten ein Spielhallen-Spiel mit Bildschirm-Scrollen erstellen. In diesem Fall müßten Ihre Scroll-Operationen unbedingt mit den Bewegungseffekten vollkommen synchronisiert werden. Wenn sich nun zum Beispiel die Aliens während des Scrollens bewegen, dann werden Ihre Hintergrundbereiche an der falschen Stelle neu gezeichnet. Das würde Ihre Anzeige völlig zerstören, und ein heilloses Durcheinander auf dem Bildschirm anrichten. Laden Sie **BEISPIEL 12.5** aus dem MANUAL-Unterverzeichnis und sehen Sie sich diesen Prozeß genauer an.

=X BOB *Gibt die x-Koordinate eines BOBs an)*

x1=X BOB(n)

Gibt die aktuelle x-Koordinate des BOBs mit der Nummer n an. Diese Koordinate wird mit Bezug zum aktuellen Bildschirm angegeben. Siehe dazu auch die Befehle Y SPRITE, X MOUSE und Y MOUSE.

=Y BOB *(Gibt die x-Koordinate eines BOBs an)*

y1=Y BOB(n)

Y BOB vervollständigt den Befehl X BOB durch die Angabe des y-Koordinate des BOBs mit der Nummer *n*. Dieser Wert wird in normalen Bildschirmkoordinaten angegeben. Siehe dazu auch die Befehle X BOB, X SPRITE, Y SPRITE, X MOUSE und Y MOUSE.

=I BOB *(Gibt das aktuelle Bild eines BOBs an)*

Image=I Bob(n)

Durch diese Funktion erhält man die Nummer des aktuellen Bildes, das gegenwärtig von dem BOB mit der Nummer n verwendet wird. Wenn das BOB nicht angezeigt wird, so erscheint dafür der Wert Null.

LIMIT BOB *(Beschränkt ein BOB auf einen rechteckigen Bildschirmabschnitt)*

LIMIT BOB [n,] x1,y1 TO x2,y2

Durch diesen Befehl sind Ihre BOBs nur in dem von den Koordinaten *x1,y1* to *x2,y2* definierten Bildschirmabschnitt sichtbar. Diese Koordinaten beziehen sich auf den aktuellen Schirm. Die x-Koordinate wird dabei auf die nächstliegende 16-Pixel- Grenze gerundet. Beachten Sie bitte, daß dieser Bereich stets breiter als Ihre BOBs sein muß, sonst erhalten Sie die Fehlermeldung Falscher Funktionsaufruf.

Wenn Sie einen Wert für den Parameter n angeben, dann gibt er die Nummer des von dieser Anweisung betroffenen BOBs an. Andernfalls wird dieser Befehl auf **alle** BOBs angewendet. Durch Eingabe von:

```
Limit Bob
```

können Sie die Begrenzung des sichtbaren Bereichs auf den gesamten Bildschirm ausdehnen.

GET BOB *(Lädt einen Bildschirmabschnitt in die Sprite-Bank)*

GET BOB [s,] i,x1,y1 TO x2,y2

Diese Anweisung ist zu dem Befehl GET SPRITE identisch. Sie stellt ein Bild aus dem aktuellen Schirm in die Sprite-Bank.

x1,y1 to *x2,y2* stellt die Koordinaten der oberen und unteren Ecke des zu erfassenden rechteckigen Bereichs dar.

i gibt die Nummer des Bildes an, das mit diesem Bereich geladen wird. s kann wahlweise eingeben werden; dieser Parameter ist dann die Nummer des Schirms, aus dem das Bild entnommen werden soll. Nähere Einzelheiten dazu finden Sie auch unter dem Befehl GET SPRITE. Hier ist ein Beispiel:

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF"
Get Sprite 1,0,64, To 320,164
Clw
Bob 1,0,0,1
```

In **BEISPIEL 12.6** finden Sie dazu ein ausführlicheres Beispiel. Hier wird ein Bild in den Speicher geladen und Sie können aus jedem beliebigen Bereich auf dem Bildschirm ein BOB in die Sprite-Bank holen. Siehe dazu auch HOT SPOT und MASK.

PUT BOB *(Die Kopie eines BOBs auf dem Bildschirm verankern)*

PUT BOB n

Dies ist das genaue Gegenteil des obengenannten Befehls GET BOB. PUT BOB stellt eine Kopie des BOBs mit der Nummer n auf seine aktuelle Position auf den Bildschirm. Das funktioniert dadurch, daß der Hintergrund des BOBs während der nächsten Vertical Blank Period nicht neu gezeichnet wird. Um die Aktualisierung der BOBs mit der Bildschirmanzeige zu synchronisieren, sollten Sie nach dieser Anweisung immer einen WAIT VBL Befehl einsetzen.

Nachdem diese Anweisung ausgeführt wurde, kann das ursprüngliche BOB ohne jede unerwünschte Nebenwirkung bewegt oder animiert werden.

PASTE BOB *(Zeichnet ein Bild aus einer Sprite-Bank auf den Bildschirm)*

PASTE BOB x,y,i

Der Befehl PASTE BOB bewirkt, daß eine Kopie des Bildes mit der Nummer i an den **Bildschirm**-Koordinaten *x,y* gezeichnet wird. Im Unterschied zu PUT BOB wird hier dieses Bild sofort auf den Bildschirm gezeichnet, und die normalen Clipping-Regeln finden Anwendung. Siehe auch PASTE ICON.

BOB OFF *(Entfernt ein BOB vom Display)*

BOB OFF [n]

Manchmal möchten Sie vielleicht bestimmte BOBs ganz und gar vom Bildschirm entfernen. Mit dem Befehl BOB OFF können Sie das BOB mit der Nummer *n* vom Bildschirm löschen und alle zugehörigen Animationen beenden. Wenn Sie keinen Wert für den Parameter *n* eingeben, werden alle BOBs durch diese Anweisung entfernt. Versuchen Sie doch einmal das folgende Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Get Sprite Palette
Bob 2,110,110,2
Wait Key
Bob Off 2
Direct
```

Das automatische Flippen von BOBs

In einer großen Reihe von Spielen muß die Hauptfigur nach links und rechts und oben und unten animiert werden. Bis jetzt mußten Sie dazu in der Sprite-Bank umgekehrte Kopien von kleinen Animationssequenzen für die Hauptfigur speichern. Da die Hauptfigur normalerweise über die beste Animation verfügt, frißt das wahnsinnig viel Speicher!

Für das Spiel RanXerox (jetzt erhältlich!), das François vor langer Zeit schrieb, erstellte er eine nette kleine Flip-Routine, durch die er nur eine Kopie seiner Hauptfigur in der Bank speichern mußte. Diese Routine wurde verbessert und in AMOS integriert.

Und wie funktioniert das nun? Angenommen, Ihre Figur läuft nach links und dann wieder nach rechts zurück. In ihrer Bank haben Sie nun nur das Bild von ihr, in dem sie nach rechts geht. Dieses rechte Bild rufen Sie nun wie gewöhnlich über die Bildnummer aus der Sprite-Bank auf. Um dieses Bild nun in der X-Achse zu drehen (und nach links laufen zu lassen), stellen Sie das Bit 15 in der Bildnummer auf 1. Nur keine Panik, das geht ganz einfach folgendermaßen:

$8000+Bildnummer

Dann zeigt

Bob 1,160,100,1

Ihre Figur, wie sie nach rechts läuft, und durch

Bob 1,160,100,$8000+1

geht sie nach links. Das gleiche Prinzip wird auch auf das Flippen in der Vertikalen angewandt. Hier wird mit dem Bit 14 gearbeitet. Zur Bildnummer addiert man nun $4000. Für das vertikale und horizontale Flippen verwendet man $C000.

Die Symmetrie ist dabei vollständig, auch der Hot-Spot des BOBs wird gedreht. Wenn wir jetzt zum Beispiel den Hot-Spot in X unter die Füße unserer Figur gelegt hatten, so wäre er dann bei der geflippten Version auch unter ihren Füßen. Seien Sie also vorsichtig, wenn Sie den Hot-Spot an die obere linke Ecke eines BOBs stellen, denn das gedrehte Bild wird auch links oben angezeigt!

Vielleicht finden Sie den Einsatz von $8000 und $C000 etwas gewöhnungsbedürftig. Wir haben daher einige spezielle Funktionen eingebaut, die eine bessere AMOS-Schnittstelle bieten:

=HREV(Bild), addiert $8000 zu dem Bild
=VREV(Bild), addiert $4000
=REV(Bild), addiert $C000

Sie können diese Funktionen folgendermaßen anstelle der Hexadezimalwerte einsetzen:

Bob 1,160,100,10

```
Bob 1,160,100,HREV(10)
Bob 1,160,100,VREV(10)
Bob 1,160,100,REV(10)
```

Damit man die BOBs auch in AMAL leicht flippen kann, haben wir die *Hexadezimal-*Bewertung eingebaut. Sie können also durch den Einsatz der Hexadezimalangabe die geflippten BOBs ganz leicht bezeichnen. Sollte das Hexadezimalformat Sie etwas abschrecken, fügen Sie einfach $8000, $4000 oder $C000 vor den Bezeichnungen in Ihren AMAL-Ketten ein. Dazu folgendes Beispiel:

Alte AMAL-Kette:

```
"Anim 0,(1,2),(2,2)(3,2)(4,2)"
```

Die neue umgekehrte Kette:

```
"Anim 0,($8000+1,2)($8000+2)($8000+3)($8000+4)"
```

oder

```
"Anim 0,($8001,2)($8002,2)($8003,2)($8004,2)"
```

Wenn Sie die Bildnummer über ein Register berechnen, versuchen Sie, nicht die Berechnung an sich zu verändern, sondern nur jeweils die Zuweisung des Registers zu dem Bild.

Alte AMAL-Kette:

```
For R0=1 To 10; Let A=R0; Next R0
```

Neue Kette:

```
For R0=1 To 10; Let A=$C000+R0; Next R0
```

Wie funktioniert die Flip-Routine?

Es ist wirklich wichtig, daß Sie das Innenleben dieser Routine verstehen, damit Sie nicht etwas von dem System erwarten, für das es sich nicht eignet.

Durch das Flip-System soll in erster Linie Speicher frei werden. Geschwindigkeit steht erst an zweiter Stelle (obwohl sie uns natürlich auch sehr wichtig ist). François mußte Kompromisse eingehen, damit es schnell und gleichzeitig einfach und leistungsstark sein kann.

Die Routine arbeitet genau in der Mitte der Bank und verbraucht keinen

zusätzlichen Speicher. Die BOBs werden während des Aktualisierens gedreht, ganz kurz bevor das BOB auf dem Bildschirm neu gezeichnet wird. AMOS sieht in der Bank nach, ob das BOB geflippt werden muß. Ist das der Fall, so wird es gedreht, und in der Bank ein Flag gesetzt. Wenn das BOB-Bild sich nicht verändert hat, wird es bei der nächsten Aktualisierung nicht noch einmal geflippt, und somit eine Menge Zeit gespart.

Wenn Sie dieses Prinzip nun verstehen, dann fällt Ihnen dabei auch eine *große* Einschränkung auf. Es ist nämlich nicht ratsam, dem gleichen Bild mehr als ein geflipptes BOB zuzuweisen. Dazu wollen wir uns das nächste Beispiel ansehen:

```
Bob 1,160,100,1
Bob 2,160,150,$8001
Bob 3,20,20,$4001
Bob 4,20,100,$C001
Update
```

Während des UPDATE-Prozesses zeichnet AMOS zuerst das BOB #1. Kein Problem, denn es ist an der richtigen Stelle. Dann BOB #2 - das muß AMOS in der X-Achse drehen. BOB #3 muß in der Y- und der X-Achse gedreht werden (damit das BOB wieder normal in X-Richtung steht). Und BOB #4 muß dann in der x-Achse gedreht werden.

Vorausgesetzt, das Bild des BOBs hat sich nicht verändert, muß AMOS BOB #1 in der X- und der Y-Achse flippen, dann BOB #2 usw.

Wie Sie sehen, müssen die BOBs, wenn sie sich bewegen, bei jeder Aktualisierung, also alle 50stel Sekunden, gedreht werden. Das funktioniert zwar, kostet aber viel Prozessorzeit und die Animation ist grauenvoll.

Die goldene Regel lautet also: geflippte BOBs nur für Objekte verwenden, die allein auf dem Bildschirm stehen (oder sicherstellen, daß normale und gedrehte Bilder nicht gleichzeitig auf dem Bildschirm dargestellt werden). Wenn Sie möchten, können Sie auch zwei derartige BOBs haben - probieren Sie es einfach aus.

Wir haben weiter oben schon erwähnt, daß dieses System für BOBs gedacht war, und hier sogar völlig automatisch ist. Da es aber direkte Auswirkungen auf die Sprite-Bank hat, können Sie es auch mit Sprites einsetzen.

Bei der Berechnung eines Hardware-Sprites sieht AMOS in der Sprite-Bank nach und entnimmt ihr dann das entsprechende Bild. Ist das Bild zu diesem Zeitpunkt geflippt, so wird auch das Hardware-Sprite ein gedrehtes Bild darstellen. Mit dieser Methode können Sie also auch geflippte Hardware-Sprites erhalten. Folgendes können Sie aber zum Beispiel nicht machen:

```
Sprite 1,200,200,$8001
```

Das Einfügen geflippter BOBs

Der Befehl PASTE BOB läßt auch gedrehte Bilder zu. Es ist zum Beispiel ein einfacher Trick, ein Bild in der Bank zu flippen, ohne ein BOB anzeigen zu müssen, indem man das geflippte Bild außerhalb des Bildschirms mit PASTE einfügt. Dazu folgendes Beispiel:

```
Paste Bob 500,500,$C000
```

Auf diese Art wird Bild 4 in der Bank gedreht, jedoch ohne angezeigt zu werden (und ziemlich schnell!).

Kollisionsabfrage

Das ist ein wichtiger Punkt, denn Sie müssen sehr vorsichtig sein, wenn Sie die Kollisionsabfrage bei geflippten BOBs durchführen.

Die Kollisionsabfrage beruht auf den Formen, die sich zu dem Zeitpunkt, wenn sie aufgerufen wird, in der Bank befinden. Wir sehen uns dazu ein Beispiel an, das nie funktioniert:

```
Bob 1,160,100,1
Do
  Bob 2,XScreen(XMouse),YScreen(YMouse),$8001
  Wait Vbl
  Exit if Bob Col(1)
Loop
```

Und warum funktioniert es nicht? Wir haben zwei geflippte Bilder mit der gleichen Definition in der Bank. Nach der Aktualisierung bleibt das Bild in der Bank umgekehrt. So greift die Anweisung zur Kollisionsabfrage auf das BOB mit der Form 1, also auf das gedrehte Bild zu, und das funktioniert nicht!

Wie wird es in die Sprite-Bank kodiert?

Es werden jeweils zwei Bits des X-Hot-Spots von jedem Bild dazu eingesetzt, das Flippen anzuzeigen (bei der Sprite-Basis +6).

Bit #15 für X-Achse:	0 für normal, 1 für umgekehrt
Bit #14 für Y-Achse:	0 für normal, 1 für umgekehrt

Vor dem Einsatz der Befehle RUN und SAVE wird die Bank in ihren Normalzustand zurückversetzt.

Das Flippen von Blöcken

Die Flip-Routine kann auch bei Blöcken eingesetzt werden.

HREV BLOCK *(Horizontales Flippen eines Blocks)*

HREV BLOCK Bild

Flippt den Block mit der Nummer *Bild* horizontal.

VREV BLOCK *(Vertikales Flippen eines Blocks)*

VREV BLOCK Bild

Flippt den Block mit der Nummer *Bild* vertikal.

AMOS-Squash

Diese Routine wurde ursprünglich für das Buchstabenspiel in einem der Lernprogramme für Kinder aus der Reihe Spielend Lernen 3 geschrieben. Jeder Buchstabe des Alphabets wurde als ein großes Sprite dargestellt. Dadurch waren im Ganzen 52 einzelne Bilder - und der erschreckende Speicherbedarf von 110 KByte - erforderlich. Da jeweils nur ein paar dieser Bilder gleichzeitig angezeigt wurden, wurde der Großteil dieses Speichers eigentlich verschwendet. Deshalb baten wir unser Programmiergenie, François Lionet, eine kleine Routine für uns zu schreiben, durch die ungenutzten Sprites in eine freie Speicherbank gepackt werden können. So konnten wir dann die gesamte Sprite-Bank zu lächerlichen 26 KByte komprimieren!

AMOS SQUASH ist jetzt als Public-Domain-Software verfügbar. Sie können es also nach Belieben in Ihren eigenen Programmen einsetzen. Es ist vor allem bei der speicherintensiven Stufenkontrolle in den Spielhallenspielen besonders nützlich. Sie kann nämlich so auf einen Bruchteil ihrer normalen Größe komprimiert werden, und am entsprechenden Punkt in Ihrem Programm sofort wieder abgerufen werden.

Der Einsatz von AMOS-Squash

Dieses Packprogramm besteht aus zwei Einzelteilen. Das erste Programm lädt eine Sprite-Bank in den Speicher und packt Ihre Bilder. Sie finden es in der Datei Squash_a_bob.AMOS.

Diese Routine können Sie einsetzen, indem Sie die folgenden, einfachen Schritte durchführen:

- Laden Sie über den Datei-Requester eine Sprite-Bank von der Diskette.
- Geben Sie die Nummer des **ersten** Sprites ein, das Sie packen möchten.
- Geben Sie die Nummer des **letzten** Sprites auf Ihrer Liste ein.
- Geben Sie die Anzahl der Farben ein, die in Ihren Sprite-Bildern eingesetzt werden.
- Wählen Sie eine neue Speicherbank für Ihre gepackten Bilder.

Nun wird die Squash-Routine ausgeführt und die von Ihnen gewählten Bilder schnell komprimiert. Nachdem die Routine fertig abgelaufen ist, erscheint eine Option, durch die Sie Ihre komprimierten Bilder auf Diskette speichern können.

Beachten Sie dabei jedoch bitte, daß **keine** Farbinformation zusammen mit diesen Bildern gespeichert wird. Sie müssen sich also die aktuelle Farbeinstellung notieren, bevor Sie weiterarbeiten.

Wenn Sie Ihre Bilder einmal gepackt haben, können Sie sie mit Hilfe von drei kleinen Prozeduren in Ihre AMOS Basic-Programme laden. Diese finden Sie in der Datei *Squash_proc.AMOS*. Mit dem **MERGE**-Befehl im AMOS Editor-Menü können sie dann in

Ihr Programm eingebaut werden.

Die Squash-Routinen sind ganz erstaunlich leicht einzusetzen. Zuerst laden Sie Ihre ursprüngliche Sprite-Bank aus dem Direkt-Modus und löschen die Bilder, die Sie gerade gepackt haben. Dazu verwenden Sie zum Beispiel die folgende Anweisung:

Del Sprite start To finish

So werden Ihre alten Bilder aus dem Speicher gelöscht, und Sie können ganz beträchtlich Speicher sparen. Nun können Sie mit dem LOAD-Befehl Ihre gepackten Sprites von der Diskette abrufen, zum Beispiel so:

Load "Bilder.abk"

Am Anfang Ihres Programms sollten Sie das Packsystem durch das Aufrufen der Prozedur PBOB_INIT initialisieren.

PBOB_INIT[Bank,Farben,Max_x,Max_y]

Dabei gibt:

Bank:	die Nummer der Bank an, die Ihre komprimierten Bilder enthält.
Farben:	die Anzahl der Farben an.
Max_x:	die maximale Breite Ihrer Bilder an.
Max_y:	die maximale Höhe Ihrer Bilder an.

Die Wirkung dieser Prozedur besteht darin, daß ein temporärer Schirm für die Squash-Utility vorbereitet wird. Nun können Ihre Bilder stets nach Bedarf einfach durch Aufrufen der PBOB-Routine wieder entpackt werden.

PBOB[Quelle,Ziel]

Quelle ist die Nummer des Bildes, das entpackt werden soll. Dies ist die ursprüngliche Bildnummer aus der Sprite-Bank.

Ziel ist die Nummer des neuen Bildes, das Sie in die Sprite-Bank stellen möchten. Wählen Sie hier die Nummer des letzten Sprites, zu der Sie eins addieren.

Nachdem Sie Ihr Bild gespeichert haben, können Sie es direkt mit den verschiedenen SPRITE- oder BOB-Befehlen animieren. PBOB können Sie sooft aufrufen, wie Sie möchten. Die gleiche Bildnummer kann also beliebig oft wiederverwendet werden.

Gegen Ende Ihres Programms lassen Sie dann die Prozedur PBOB_END noch aufrufen. So wird der verborgene Schirm, der durch PBOB_INIT erzeugt wurde, wieder gelöscht.



13: Die Objektsteuerung

In diesem Kapitel wollen wir Ihnen zeigen, wie die Objekte, die Sie mit den Sprite- und BOB-Befehlen erzeugt haben, aus einem AMOS Basic-Programm heraus gesteuert werden können. Unter anderem wollen wir hier auch die Themen Kollisionsabfrage und Einsatz von Maus-Cursor und Joystick besprechen.

Der Mauszeiger

Der Maus-Cursor bietet dem Spieleprogrammierer eine wertvolle Alternative zum normalen Joystick. Mit dem Befehl CHANGE MOUSE können Sie die Maus durch ein Bild aus der aktuellen Sprite-Bank ersetzen. Außerdem gibt es eine Reihe von Anweisungen, durch die Sie jederzeit Position und Status dieser Maus bestimmen können. Diese Anweisungen umfassen die Befehle X MOUSE, Y MOUSE und MOUSE KEY.

HIDE *(Entfernt den Mauszeiger vom Bildschirm)*

HIDE [ON]

Dieser Befehl entfernt den Mauszeiger völlig vom Bildschirm. Das System zählt intern, wie oft Sie diese Funktion aufgerufen haben. Bevor der Zeiger wieder auf dem Bildschirm erscheint, müssen Sie auch die Anweisung SHOW entsprechend oft rufen.

Sie erhalten über die Eingabe von HIDE ON eine weitere Version dieser Anweisung. Hier wird das Aufrufen der Funktion nicht gezählt, und die Maus ständig versteckt, ganz egal, wie oft Sie den Befehl SHOW eingeben.

Beachten Sie, daß nur der Mauszeiger durch HIDE unsichtbar wird. Dieser Befehl hat keine Auswirkungen auf die anderen AMOS-Befehle. Sie können also die Koordinaten der Maus nach wie vor mit den Funktionen X MOUSE und Y MOUSE abfragen.

SHOW *(Aktiviert den Mauszeiger)*

SHOW [ON]

Mit dieser Anweisung wird der Mauszeiger nach einem HIDE-Befehl auf dem Bildschirm wieder angezeigt. Als Default wird gezählt, wie oft vorher der HIDE- Befehl aufgerufen wurde. Wenn die Anzahl der SHOW-Anweisungen geringer als die der HIDE-Befehle ist, bleibt der Zeiger unsichtbar. Um diese Zahl zu ignorieren und den Mauszeiger sofort wieder anzuzeigen, können Sie stattdessen den Befehl SHOW ON einsetzen.

CHANGE MOUSE *(Verändert die Form des Mauszeigers)*

CHANGE MOUSE m

Mit dieser Anweisung können Sie jederzeit die Form der Maus verändern. Für die Form des Mauszeigers gibt es drei Standardmöglichkeiten. Ihnen können die Nummern 1 - 3 wie folgt zugewiesen werden:

M	Form
1	Mauspfeil (Default)
2	Fadenkreuz
3	Uhr

Wenn Sie für *m* einen Wert eingeben, der größer als 3 ist, dann nimmt AMOS an, daß Sie auf ein in der Sprite-Bank gespeichertes Bild Bezug nehmen. Die Nummer dieses Bildes wird durch den Ausdruck *l=m-3* bestimmt. So wird das Bild mit der Nummer 1 mit dem Wert 4 installiert, und das Bild mit der Nummer 2 mit dem Wert 5 geladen.

Damit Sie diese Option nutzen können, muß Ihr Sprite-Bild genau 16 Pixel breit sein und darf aus nicht mehr als vier Farben bestehen. Die Höhe Ihres Bildes ist jedoch nicht begrenzt. Siehe auch die Befehle HIDE, SHOW, X MOUSE, Y MOUSE, MOUSE KEY und LIMIT MOUSE.

=MOUSE KEY *(Gibt den Status der Maustasten an)*

k=MOUSE KEY

Die Funktion MOUSE KEY ermöglicht es Ihnen, schnell zu überprüfen, ob eine oder mehrere der Maustasten gedrückt wurden. Sie erhalten durch diese Abfrage ein Bitmuster, das den Status der Maustasten widerspiegelt.

Dieses Bitmuster hat folgendes Format:

Bit 0	Auf 1 setzen, wenn linke Maustaste gedrückt wurde, sonst Null
Bit 1	Status der RECHTEN Maustaste, im selben Format angegeben
Bit 2	Status der DRITTEN Maustaste, falls vorhanden

Hier ist ein Beispiel:

```
Curs Off
Do
  Locate 0,0
   M=Mouse Key: Print "Bitmuster";Bin$(M,8);"Nummer",M
Loop
```

=MOUSE CLICK *(Überprüft das Anklicken)*

c=MOUSE CLICK

Die Funktion MOUSE CLICK überprüft, ob der Anwender eine Maustaste angeklickt hat. Sie gibt ein Bitmuster in folgendem Format aus:

Bit 0	Nur einmal ausgeführte Abfrage der LINKEN Taste
Bit 1	Nur einmal ausgeführte Abfrage der RECHTEN Taste
Bit 2	Nur einmal ausgeführte Abfrage der DRITTEN Taste, falls vorhanden

Diese Abfrage wird nur einmal ausgeführt, und nur das Ergebnis lautet nur dann eins, wenn die Maustaste gerade gedrückt wurde. Nachdem die Abfrage einmal durchgeführt worden ist, werden diese Bits automatisch auf Null zurückgesetzt. Deshalb können Sie so den Status jedes Mal nur auf einen einzigen Tastendruck überprüfen. Hier ist ein Beispiel dazu:

```
Curs Off
Do
  M=Mouse Key
  If M<>0 Then Print "Bitmuster";Bin$(M,8);"Nummer",M
Loop
```

=X MOUSE= *(Angeben/Bestimmen der x-Koordinate des Mauszeigers)*

x1=X MOUSE

X MOUSE gibt die aktuelle x-Koordinate des Mauszeigers im *Hardware*-Format an.

Mit dieser Funktion können Sie die Maus auch zu einer bestimmten Bildschirmposition **bewegen**. Das erreichen Sie, indem Sie der Maus wie einer Basic-Variablen einen Wert zuweisen. Zum Beispiel:

```
X Mouse=150 : Rem Bewegt Maus zu Hardware-Koordinate 150, Y-Koord ist nicht davon
betroffen
```

=Y MOUSE= *(Angeben/Bestimmen der y-Koordinate des Mauszeigers)*

y1=Y MOUSE

Diese Funktion weist die y-Koordinate des Mauszeigers aus. Wie bei X MOUSE stellt das Ergebnis *Hardware*-Koordinaten und nicht die gebräuchlicheren Bildschirm-Koordinaten dar. Mit Y MOUSE kann man den Mauszeiger ebenfalls auf dem Bildschirm positionieren. Laden Sie dazu einfach folgendermaßen die neue y- Koordinate in Y MOUSE:

```
Y Mouse=100
```

Wenn Sie die Funktionen X MOUSE und Y MOUSE an einem Beispiel sehen möchten, laden Sie einfach **BEISPIEL 13.1** aus dem MANUAL-Unterverzeichnis.

LIMIT MOUSE *(Beschränkt die Maus auf einen Bildschirmabschnitt)*

LIMIT MOUSE x1,y1 TO x2,y2

Durch diese Anweisung werden die Maus-Bewegungen auf den durch die Hardware-Koordinaten *x1,y1* und *x2,y2* definierten rechteckigen Bereich beschränkt. *x1,y1* stellt die linke obere Ecke und *x2,y2* den diagonal gegenüberliegenden Punkt dar. Anders als bei dem Befehl LIMIT BOB ist die Maus in diesem Bereich völlig gefangen und kann nicht herausbewegt werden.

Laden Sie **BEISPIEL 13.2** aus dem MANUAL-Unterverzeichnis und sehen Sie sich eine Demonstration dieses Befehls an. Wenn Sie die Maus wieder über den gesamten Bildschirmbereich bewegen möchten, geben Sie den Befehl einfach nochmals ohne Angabe der Parameter ein.

Limit Mouse

Abfragen des Joysticks

AMOS-Basic enthält sechs Funktionen, durch die Sie sofort die Bewegungen eines angeschlossenen Joysticks überprüfen können.

=JOY *(Abfragen des Joysticks)*

d=JOY(j)

Diese Funktion gibt eine Binärzahl an, die den aktuellen Status eines an den Port mit der Nummer *j* angeschlossen Joysticks darstellt. Normalerweise wird Ihr Joystick an die linke Buchse (Nummer 1) angeschlossen. Sie können jedoch auch die Maus von der rechten Buchse entfernen und durch einen Joystick ersetzen. Jetzt erhalten Sie über den Port mit der Nummer 0 Zugang.

Der Zustand des Joysticks wird durch Überprüfen des ausgewiesenen Musters aus binären Bits deutlich. Jedes Bit zeigt an, ob der Anwender eine bestimmte Aktion durchgeführt hat. Wenn ein Bit auf eins gestellt ist, dann ist die entsprechende Abfrage positiv und der Joystick wurde in diese Richtung bewegt.

Hier sind nun die verschiedenen Bits und ihre Bedeutung aufgeführt:

Bit Nummer	Bedeutung
0	Joystick nach oben bewegt
1	Joystick nach unten bewegt
2	Joystick nach links bewegt
3	Joystick nach rechts bewegt
4	Feuerknopf gedrückt

Nur keine Angst, wenn Sie mit diesen Binärangaben nicht vertraut sind. Sie können auch jede der Richtungen einzeln mit den Befehlen JLEFT, JRIGHT, JUP, JDOWN und FIRE abfragen. In **BEISPIEL 13.3** finden Sie ein Beispiel für diesen Befehl.

=JLEFT *(Abfragen der Bewegung des Joysticks nach links)*

x=JLEFT(j)

JLEFT weist einen Wert von -1 (True = richtig) aus, wenn der Joystick in Port *j* nach links bewegt wurde. Ansonsten wird 0 (False = falsch) angezeigt. Die Joystick-Buchsen sind von links nach rechts numeriert, dabei wird bei Null begonnen. Deshalb wird die Default-Joystick-Buchse als Port Nummer 1 bezeichnet. Hier ist ein Beispiel:

```
Do
  If Jleft(1) The Print "LINKS"
Loop
```

=JRIGHT *(Abfragen der Bewegung des Joysticks nach rechts)*

x=JRIGHT(j)

JRIGHT weist einen Wert von -1 (True = richtig) aus, wenn der Joystick in Port *j* nach rechts bewegt wurde, sonst wird 0 (False = falsch) angezeigt.
Siehe auch die Befehle JLEFT, JUP und JDOWN.

=JUP *(Abfragen der Bewegung des Joysticks nach oben)*

x=JUP(j)

JUP weist einen Wert von -1 (True = richtig) aus, wenn der Joystick in Port *j* nach oben bewegt wurde, sonst wird 0 (False = falsch) angezeigt. Siehe auch die Befehle JRIGHT, JLEFT und JDOWN.

Das Abfragen von BOB-Kollisionen

BOB COL *(Erfaßt Kollisionen zwischen zwei Blitter-Objekten)*

c=BOB COL(n[,s TO e])

Die Funktion BOB COL überprüft das BOB mit der Nummer n auf eine Kollision mit einem anderen BOB. Wenn eine Kollision festgestellt wird, dann lautet der in c ausgegebene Wert -1 (richtig), ansonsten wird 0 (falsch) angezeigt.

Normalerweise ermittelt dieser Befehl alle Kollisionen. Sie können jedoch wahlweise auch über die Bereichsparameter s bis e eine Reihe von BOBs eingeben, die untersucht werden sollen. Der Status der einzelnen BOBs kann mit der COL- Anweisung untersucht werden. Siehe dazu auch **BEISPIEL 13.5**.

Kollision zwischen BOBs und Sprites

Mit AMOS-Basic müssen Sie sich nicht nur darauf beschränken, Kollisionen zwischen der gleichen Art von Objekten abzufragen. Sie können vielmehr jede beliebige Kombination von BOBs und Sprites auf dem Bildschirm überprüfen.

SPRITEBOB COL *(Stellt Kollision zwischen Sprites und BOBs fest)*

c=SPRITEBOB COL(n[,s TO e])

Diese Funktion überprüft, ob eine Kollision zwischen dem **Sprite** mit der Nummer *n* und einem oder mehreren **BOBs** stattgefunden hat. Der für c ausgewiesene Wert beträgt entweder -1 (wenn eine Kollision festgestellt wurde) oder 0, wenn keine Kollisionen entdeckt wurden. Die Anfangs- und Endpunkte geben an, daß nur Kollisionen mit den BOBs im Bereich der Nummern s bis e untersucht werden. Weitere Einzelheiten dazu können Sie unter dem COL-Befehl finden.

Achtung! Das Feststellen von Kollisionen zwischen einem Sprite und einem BOB ist nur auf Bildschirmen mit niedriger Auflösung (Lowres) möglich. Im Hires-Modus sind die Pixel-Größen, die für Sprites und BOBs verwendet werden, völlig unterschiedlich, und daher sind die Ergebnisse dieser Funktion äußerst unzuverlässig.

BOBSPRITE COL *(Stellt eine Kollision zwischen BOBs und Sprites fest)*

c=BOBSPRITE COL(n[,s TO e])

Diese Funktion überprüft, ob eine Kollision zwischen einem bestimmten BOB und mehreren Sprites stattgefunden hat. Der für c ausgewiesene Wert beträgt -1, wenn die Abfrage erfolgreich war, sonst wird 0 angezeigt. Die Anfangs- und Endpunkte geben die Gruppe von Sprites an, die untersucht werden soll. Wenn diese Parameter nicht

=JDOWN *(Abfragen der Bewegung des Joysticks nach unten)*

x=JDOWN(j)

JDOWN weist einen Wert von -1 (True = richtig) aus, wenn der Joystick in Port *j* nach unten bewegt wurde, sonst wird 0 (False = falsch) angezeigt. Siehe auch die Befehle JRIGHT, JLEFT und JUP.

=FIRE *(Gibt den Status des Feuerknopfes an)*

x=FIRE(j)

FIRE gibt durch einen Wert von -1 (True = richtig) an, daß der Feuerknopf am Joystick in Port *j* gedrückt wurde. Ansonsten wird 0 (False = falsch) angezeigt. Siehe auch die Befehle JUP, JDOWN, JLEFT, JRIGHT und JOY.

Die Kollisionsabfrage

SPRITE COL *(Erfaßt Kollisionen zwischen zwei Hardware-Sprites)*

c=SPRITE COL(n[,s TO e])

Dies ist ein einfacher Weg, zu testen ob zwei oder mehr Sprites auf dem Bildschirm kollidiert sind. Die Zahl *n* bezieht sich auf ein aktives Hardware-Sprite. Sie wollen nun feststellen, ob dieses Hardware-Sprite in eine Kollision verwickelt war. Wenn tatsächlich eine Kollision stattgefunden hat, wird ein Wert von -1 (True = richtig) ausgewiesen, sonst lautet das Ergebnis 0 (False = falsch).

In der Standardausführung (ohne Parameter) fragt dieser Befehl alle Sprites auf Kollisionen ab. Mit der erweiterten Form der Anweisung können Sie jedoch eine bestimmte Gruppe von Sprites untersuchen:

c=SPRITE COL n,s TO e

Diese Anweisung überprüft, ob es zwischen dem Sprite *n* und den im Bereich zwischen *s* und *e* liegenden Sprites zur Kollision gekommen ist. Wenn eine Kollision festgestellt wird, dann können Sie die Nummern der betroffenen Sprites mit der COL-Funktion abfragen.

Beachten Sie, daß Sie diese Funktion nur einsetzen können, wenn Sie zuvor mir dem Befehl MAKE MASK eine Sprite Maske erzeugt haben, sonst kann die Kollision nicht festgestellt werden. Sie können in **BEISPIEL 13.4** ein ausführliches Beispiel für diesen Befehl finden.

eingeben werden, dann fragt diese Funktion alle gegenwärtig aktiven Sprites ab.

Achtung! Die Funktion BOBSPRITE COL kann nur bei niedriger Auflösung (Lowres) eingesetzt werden. Wenn Sie versuchen, sie in Verbindung mit dem Hires-Modus anzuwenden, erhalten Sie unzuverlässige Ergebnisse. Eine ausführliche Demonstration dieses Befehls finden Sie in **BEISPIEL 13.6** im MANUAL-Unterverzeichnis.

=COL

(Überprüft den Status eines Sprites oder BOBs nach einer Kollisionsabfrage)

c=COL(n)

Das COL-Feld enthält den Status aller Objekte, die vorher durch die Funktionen zur Kollisionsabfrage überprüft wurden.

Jedes Objekt, das Sie untersucht haben, wird mit einem Element in diesem Feld in Bezug gesetzt. Wenn eine Kollision mit dem Objekt mit der Nummer *n* festgestellt wurde, dann wird dieses Element mit dem Wert -1 geladen, sonst erhält es den Wert 0. Die Numerierung ist ganz einfach: das erste Element im Feld gibt den Status des Objektes mit der Nummer 1 an, das zweite stellt Objekt Nummer 2 dar usw. Siehe dazu auch **BEISPIEL 13.7**.

Die zweite und noch leistungsstärkere Version der Funktion =COL ist, es mit einem negativen Wert aufzurufen. =COL(-n) ergibt als Ergebnis das erste Objekt, welches mit dem Objekt n eine Kollision gehabt hat. Um die eventuelle weitere, an der gleichen Kollision beteiligten Objekte zu ermitteln, muß =COL nochmals aufgerufen werden, jeweils mit dem vorigen Wert+1! Hierzu ein Beispiel:

```
B=Bob Col(0)
BB=0
Repeat
   BB=Col(-(BB+1))
   If BB
      Print "Bob ";bb;" hat mit Bob 0 kollidiert !"
   Endif
Until BB=0
```

Bitte beachten Sie, daß mit dieser Methode die Kollisionen des Bobs 0 nicht festgestellt werden können, hierzu müssen Sie auf die erste Variante der =COL Funktion zurückgreifen.

Wenn Sie mit den Anweisungen SPRITE COL oder BOB SPRITE COL arbeiten, handelt es sich bei den Objekten um Hardware-Sprites, ansonsten sind es Blitter-Objekte. Um Verwirrung zu vermeiden, rufen Sie diesem Befehl am besten sofort nach der entsprechenden Anweisung zur Kollisionsabfrage auf.

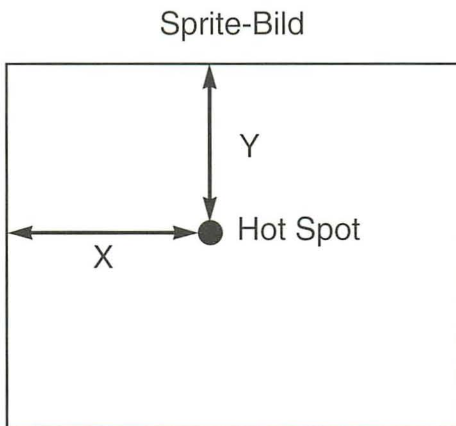
HOT SPOT *(Bestimmt den Hot Spot für ein Bild in der Sprite-Bank)*

HOT SPOT Bild,x,y
HOT SPOT Bild,p

Dieser Befehl bestimmt den Hot Spot eines in der aktuellen Sprite-Bank gespeicherten Bildes. Der Hot Spot des Objektes dient als Referenzpunkt für alle Koordinatenberechnungen. Es gibt zwei Versionen dieser Anweisung.

HOT SPOT Bild,x,y

Die *x,y*-Koordinaten werden von der oberen linken Ecke des Bildes aus gemessen. Diese Koordinaten werden dann zu den Sprite- oder BOB-Koordinaten addiert, um ein Objekt ganz exakt auf dem Bildschirm zu plazieren.



Beachten Sie bitte, daß es vollkommen erlaubt ist, wenn der Hot Spot außerhalb des eigentlichen Bildes liegt.

HOT SPOT Bild,p

Dies ist eine Kurzform der Anweisung, durch die der Hot Spot auf eine von neun vorher definierten Positionen verschoben wird. Diese Positionen werden im nachstehenden Diagramm aufgezeigt, wobei der Mittelpunkt des Bildes durch den Wert $11 dargestellt wird.

$00	$10	$20
$01	$11	$21
$02	$12	$22

Siehe hierzu auch **BEISPIEL 13.8**.

MAKE MASK *(Erstellt eine Maske um ein Bild zur Feststellung einer Kollision)*

MAKE MASK [n]

MAKE MASK definiert eine Maske um das Bild mit der Nummer *n* in der Sprite- Bank. Diese Maske wird dann von allen Anweisungen zur Kollisionsabfrage in AMOS-Basic eingesetzt. Sie sollten daher für jedes Objekt, das Sie überprüfen möchten, eine Maske erstellen. Wenn Sie die Bildnummer *n* nicht eingeben, dann wird für jedes Bild in der Sprite-Bank eine Maske generiert. Das kann jedoch ein Weilchen dauern.

Sie sollten dabei nicht vergessen, daß die Masken automatisch generiert werden, sobald das erste Mal auf dem Bildschirm gezeichnet wird. Das kann zu einer beträchtlichen Verzögerung beim Ablauf Ihres Programms führen, deshalb lohnt es sich auf alle Fälle, MAKE MASK während der Initialisierungs-Prozedur ausdrücklich aufzurufen.

Kollisionen mit rechteckigen Blöcken

AMOS-Basic enthält eine Reihe von Funktionen, durch die Sie schnell nachsehen können, ob ein Sprite oder BOB in einen rechteckigen Bereich auf dem Bildschirm eingetreten ist.

Diese *Bildschirmzonen* sind besonders wirksam zur Kollisionsabfrage in Geschicklichkeitsspielen wie Arkanoid, da dann jedem Block eine eigene Bildschirmzone zugewiesen werden kann. Sie können auch die Zonen für die Tasten und Schalter verwenden, die Sie für die Kontrollfelder und Dialogboxen brauchen.

RESERVE ZONE *(Reserviert Platz für eine Abfragezone)*

RESERVE ZONE [n]

RESERVE ZONE weist ausreichend Speicherplatz für genau *n* Abfragezonen zu. Sie sollten diesen Befehl immer einsetzen, bevor Sie mit SET ZONE eine Zone definieren.

Die einzige Begrenzung der Anzahl der Zonen besteht in der Größe des verfügbaren Speicherplatzes. Es ist also gut möglich, daß Sie in einem Ihrer Programme Hunderte oder sogar Tausende von Zonen definieren.

Um die aktuellen Zonendefinitionen zu löschen und den Speicherplatz dem Hauptprogramm wieder zuzuführen, geben Sie RESERVE ZONE einfach ohne Angabe der Parameter ein.

SET ZONE *(Definiert eine Zone, die überprüft werden soll)*

SET ZONE z,x1,y1 TO x2,y2

Definiert eine rechteckige Zone, die danach mit den verschiedenen ZONE-Befehlen überprüft werden kann. *z* stellt die Nummer der Zone dar, die erstellt werden soll. *x1,y1*

und *x2,y2* geben die Koordinaten der linken oberen Ecke und der rechten unteren Ecke des Rechtecks ein.

Bevor Sie diese Anweisung jedoch einsetzen, müssen Sie mit RESERVE ZONE etwas Speicherplatz für Ihre Zonen reservieren.

Siehe auch die Befehle ZONE, RESET ZONE, RESERVE ZONE und ZONE$.

=ZONE *(Weist eine Zone an den vorgegebenen Bildschirmkoordinaten aus)*

t=ZONE([s],x,y)

Durch die Anweisung ZONE erhalten Sie die Nummer der Bildschirmzone bei den Grafik-Koordinaten *x,y*. Normalerweise beziehen sich die Koordinaten auf den aktuellen Schirm, und Sie können auch wahlweise die Schirmnummer s als Parameter bei dieser Funktion eingeben.

Nachdem ZONE aufgerufen wurde, enthält *t* entweder die Nummer der Zone bei den angegebenen Koordinaten oder den Wert 0 (falsch).

Beachten Sie bitte, daß ZONE nur die erste Zone an diesen Koordinaten ausweist. Alle weiteren Zonen, die in diesem Bereich liegen, können nicht erfaßt werden.

Um Ihnen zu zeigen, welche Wirkung dieser Befehl hat, sind im MANUAL-Unterverzeichnis zwei Beispiele mit der Bezeichnung **BEISPIEL 13.9** und **BEISPIEL 13.10** enthalten. Sie können diese Beispiele nach Belieben verändern und in Ihren Spielen einsetzen.

Sie können diese Funktion auch in Verbindung mit den Funktionen X BOB und Y BOB einsetzen, um festzustellen, ob ein BOB in eine bestimmte Bildschirmzone eingetreten ist. Das können Sie durch Verwendung des folgenden Codes erreichen:

```
X=Zone(X Bob(n),Y Bob(n))
```

Siehe auch die Befehle HZONE, SET ZONE, RESET ZONE, X BOB und Y BOB.

=HZONE *(Gibt die Zone an den angegebenen Hardware-Koordinaten an)*

t=HZONE([s],x,y)

Der Befehl HZONE stimmt fast ganz genau mit ZONE überein, davon abgesehen, daß die Bildschirmposition jetzt in Hardware-Koordinaten angegeben wird. Sie können diese Funktion daher einsetzen um festzustellen, wann ein Hardware-Sprite in eine Ihrer Bildschirmzone eintritt. Hier ist ein Beispiel dazu:

```
X=Hzone(X Sprite(n),Y Sprite(N))
```

Eine Demonstration dieses Befehls finden Sie in **BEISPIEL 13.11**. Siehe auch die Befehle ZONE, MOUSE ZONE, SET ZONE, RESERVE ZONE und ZONE$.

=MOUSE ZONE *(Überprüft, ob der Mauszeiger in eine Zone eingetreten ist)*

x=MOUSE ZONE

Die Funktion MOUSE ZONE gibt die Nummer der Bildschirmzone an, in der sich der Mauszeiger gegenwärtig befindet. Sie entspricht der Zeile:

```
X=Hzone(X mouse,Y mouse)
```

Siehe auch ZONE, HZONE, SET ZONE und ZONE$.

RESET ZONE *(Löscht eine Zone)*

RESET ZONE [z]

Mit diesem Befehl können Sie jede der Zonen, die Sie mit SET ZONE erzeugt haben, dauerhaft außer Kraft setzen. Wenn Sie einen Wert für den Parameter z eingeben, dann ist davon nur die Zone mit dieser Nummer betroffen, ansonsten wirkt dieser Befehl auf alle Zonen. Beachten Sie bitte, daß RESET ZONE nur die Definition der Zonen löscht und nicht den durch RESERVE ZONE zugewiesenen Speicher aufhebt.

Die Wertigkeit der BOBs

PRIORITY ON/OFF *(Wechsel der Wertigkeits-Modi)*

PRIORITY ON/OFF

Jedem BOB wird ein Wert zwischen 0 und 63 zugewiesen, der seine Wertigkeit darstellt. AMOS-Basic kann anhand dieser Nummer entscheiden, in welcher Reihenfolge die Objekte auf dem Bildschirm dargestellt werden sollen. In der Regel wird jedes BOB mit der höchsten Wertigkeit stets vor den Objekten mit niedrigerer Wertigkeit angezeigt. Die Wertigkeit entspricht genau der Nummer eines BOBs.

Diesen Umstand sollten Sie nicht vergessen, wenn Sie Ihren BOBs Nummern zuweisen. Die Wahl der Nummer kann weitreichende Auswirkungen auf das Erscheinungsbild Ihrer Objekte auf dem Bildschirm haben.

Zusätzlich zu dem Standardsystem können Sie die BOBs auch nach ihrer Position auf dem Bildschirm anordnen. PRIORITY ON weist den BOBs mit den höchsten y-Koordinaten die größte Wertigkeit zu. So können Sie dann zum Beispiel die hilfreiche Illusion einer Perspektive in Ihren Spielen erzeugen. Sehen Sie sich doch einmal das folgende Beispiel an:

```
Load "AMOS_DATA/Sprites/Monkey_right.abk" : Cls : Flash Off : Get Sprite Palette
Double Buffer
```

```
Priority Off : Rem Normalen Modus einstellen
Bob 1,160,100,2 : Channel 3 To Bob 3
Amal 2," Loop: M 320,0,320 ; M -320,0,320 ; Jump Loop"
Amal 3," Loop: M -320,0,320; M 320,0,320 ; Jump Loop"Amal On
Wait Key
Priority On : Rem Y-Modus einstellen
Wait Key
```

Normalerweise bewegen sich beide BOBs unten am Objekt in der Mitte vorbei. Wenn Sie nun das Wertigkeitssystem durch Aufrufen von PRIORITY ON verändern, werden diese BOBs jetzt nach steigenden y-Koordinaten angeordnet. So bewegt sich BOB drei nun über BOB eins, während BOB zwei gleichzeitig dahinter weggeht.

HINWEIS: Es ist meistens am besten, den Hot Spot eines Sprites an seiner Grundlinie zu positionieren. Das beruht darauf, daß sich die y-Koordinate, auf die dieser Befehl zugreift, auf die Position des Hot Spots auf dem Bildschirm bezieht. Beachten Sie bitte auch, daß Sie mit der Anweisung PRIORITY OFF die Wertigkeit wieder auf den Normalzustand zurücksetzen können.

PRIORITY REVERSE ON/OFF

(Verändert die Reihenfolge, in der BOBs auf dem Bildschirm angezeigt werden)

PRIORITY REVERSE ON
PRIORITY REVERSE OFF

Der Befehl PRIORITY REVERSE ON dreht die gesamte Wertigkeitstabelle der BOBs um. Das bedeutet, daß BOB Nummer 1 als erster, vor allen anderen BOBs angezeigt wird, dann Nummer 2 usw. Diese Wertigkeitsliste ist zu STOS kompatibel.

Diese Anweisung verfügt über eine weitere angenehme Eigenschaft, wenn Sie sie in Verbindung mit PRIORITY ON einsetzen. Die BOBs werden dann nämlich nicht mehr von OBEN nach UNTEN sortiert, sondern von UNTEN nach OBEN. So wird das höchste BOB auf dem Bildschirm vor den anderen angezeigt.

Sonstige Befehle

UPDATE

(Verändert die automatische Aktualisierung der Sprites/BOBs)

UPDATE [ON/OFF]

Normalerweise werden alle Objekte, die Sie auf den Bildschirm zeichnen, neu angezeigt, ganz egal, ob sie animiert oder bewegt werden. Dieser Vorgang kann zeitweise durch den Befehl UPDATE OFF unterbunden werden. Wenn die Aktualisierung nicht aktiv ist, haben die SPRITE-, BOB- und AMAL-Anweisungen keine offensichtliche Wirkung. Tatsächlich werden aber Ihre Animationen korrekt durchgeführt - nur die Ergebnisse erscheinen nicht auf dem Bildschirm! Sie können das Neuzeichnen aber jederzeit durch Aufrufen des UPDATE-Befehls wieder hervorrufen. Hier sind nun die drei verschiedenen Formen der UPDATE-Anweisung:

UPDATE OFF

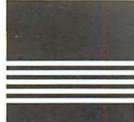
Schaltet die automatische Aktualisierung von Sprites und BOBs aus. Alle Bewegungen oder Animationen scheinen nicht stattzufinden.

UPDATE

Zeichnet alle Sprites, die ihre ursprüngliche Position verlassen haben, neu.

UPDATE ON

Die Aktualisierung der Sprites wird jetzt wieder ganz normal durchgeführt. Siehe dazu **BEISPIEL 13.12**. Siehe auch die Befehle UPDATE EVERY, SYNCHRO, SPRITE UPDATE und BOB UPDATE.



14: AMAL

Damit Sie die glatten Bewegungen erzeugen können, die für eine gutes Spielhallen-Spiel so unerläßlich sind, muß jedes Objekt auf dem Bildschirm dutzendmal pro Sekunde bewegt werden. Damit hat man sogar wenn man im Maschinencode programmiert, ziemlich zu kämpfen und es übersteigt selbst die Fähigkeiten der schnellsten Basic-Version bei weitem.

AMOS umgeht dieses Problem, denn es beinhaltet eine leistungsstarke Programmiersprache zur Animation, die von Ihren Basic-Programmen völlig unabhängig ist. Auf diese Art können Sie Animationseffekte mit einer Geschwindigkeit hervorrufen, die Sie im normalen Basic niemals erreichen könnten.

Diese Programmiersprache, die **AM**OS **A**nimation **L**anguage (AMAL) ist einzigartig. Mit ihr können Sie so ziemlich alles, vom Sprite bis zu ganzen Schirmen, mit unwahrscheinlicher Geschwindigkeit animieren. Durch den Einsatz von Interrupts können bis zu 16 AMAL-Programme gleichzeitig ausgeführt werden.

Jedes Programm steuert die Bewegungen eines Objektes auf dem Bildschirm. Sie können die Objekte in komplexen, vorgegebenen Angriffsmustern bewegen, die über ein eigenes Editor-Zusatzprogramm erzeugt werden. Falls erforderlich können Sie die Objekte aber auch direkt über die Maus oder den Joystick steuern.

Wie unwahrscheinlich vielseitig das AMAL-System ist, glaubt man eigentlich erst, wenn man es gesehen hat. Laden Sie deshalb **BEISPIEL 1** aus dem MANUAL-Unterverzeichnis und sehen Sie sich eine umfassende Demonstration an.

Das Prinzip von AMAL

AMAL ist im Grunde nur eine einfache Basic-Version, die sorgfältig optimiert wurde, um die größtmögliche Geschwindigkeit zu erreichen. Wie bei Basic gibt es Anweisungen zur Programmsteuerung (Jump), zum Treffen von Entscheidungen (If) und Wiederholen von Codeabschnitten in Schleifen (For...Next). Aber der durchschlagende Effekt wird erst ersichtlich, wenn ein AMAL-Programm abläuft. Dann werden nämlich die Befehle nicht nur blitzschnell ausgeführt, sondern alle AMAL-Programme werden vor der Durchlaufzeit **kompiliert**.

AMAL-Befehle werden über kurze Schlüsselworte eingegeben, die aus einem oder mehreren Großbuchstaben bestehen. Auf diese Art können Sie Ihre AMAL-Anweisungen etwas auspolstern und sie so etwas lesbarer gestalten. So kann für den Befehl M zum Beispiel Move oder die Anweisung L Let eingegeben werden.

AMAL-Anweisung können durch fast alle nicht verwendeten Zeichen voneinander abgetrennt werden. Eine Ausnahme ist hier der Doppelpunkt ":"; er kann nicht verwendet werden, weil er für die Definition von Sprungmarken (Labels) eingesetzt wird. Wir würden Ihnen raten, hier den Strichpunkt ";" zu verwenden, so können Sie eventuelle AMAL-Probleme am besten vermeiden.

Sie können Ihre AMAL-Programme auf zwei verschiedene Arten erstellen. Zum einen können Sie Ihre Animationssequenzen mit dem AMAL-Zusatzprogramm erzeugen und sie dann in einer Speicherbank speichern. Oder Sie können Ihre Animationen direkt in AMOS-Basic über eine AMAL-Anweisung definieren. Das allgemeine Format dieser Anweisung lautet folgendermaßen:

AMAL n,a$

n stellt die Identifikationsnummer Ihres neuen AMAL-Programms dar. Als Default werden alle Programme den entsprechenden Hardware-Sprites zugewiesen. Also steuert das erste AMAL-Programm das Sprite mit der Nummer eins, das zweite das Sprite Nummer zwei usw. Sie können diese Auswahl jederzeit mit einem eigenen CHANNEL-Befehl verändern. *a$* ist eine Zeichenkette, die eine Reihe von AMAL- Anweisungen enthält, die in Ihrem Programm ausgeführt werden sollen. Hier ist ein einfaches Beispiel:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk": Rem Lädt einige Beispiel- Sprites
Get Sprite Palette
Sprite 8,130,50,1 : Rem Bringt ein Sprite auf den Bildschirm
Rem Definiert ein kleines AMAL-Programm
Amal 8,"S: M 300,200,100 ; M -300,-200,-100 J S"
Amal On 8 : Rem Aktiviert AMAL-Programm Nummer acht
Direct
```

Durch den DIRECT-Befehl bringt Sie das Programm umgehend zurück in den Direktmodus. Versuchen Sie an diesem Punkt ein paar Basic-Befehle einzugeben. Sie können sehen, wie der Bewegungsablauf dessen ungeachtet weiterläuft, ohne Auswirkungen das übrige AMOS-System. Beachten Sie auch, daß wir Sprite 8 eingesetzt haben, um die Verwendung eines berechneten Sprites zu erzielen. Alle berechneten Sprites von 8 bis 15 werden vom AMAL-System automatisch den entsprechenden Kanalnummern zugewiesen. Deshalb sind besondere Initialisierungs-Prozeduren garnicht erforderlich. Es ist am sichersten, wenn Sie sich in Ihren Programmen ausschließlich auf berechnete Sprites beschränken. Außer Sie wollen die Anzahl der Hardware-Sprites begrenzen. Beachten Sie bitte auch, wie wir das AMAL-Programm mit dem Befehl AMAL ON aktiviert haben. Diese Anweisung hat folgendes Format:

AMAL ON [Prog]

Prog gibt hier die Nummer eines bestimmten AMAL-Programms an, das Sie starten möchten. Wenn Sie hier keinen Wert eingeben, werden alle Ihre AMAL-Programme gleichzeitig ausgeführt.

Das AMAL-Tutorial

Und jetzt wollen wir mit unserer Führung durch das AMAL-System beginnen. Sie können sich hier langsam mit der Funktionsweise der AMAL-Programme vertraut machen, ohne sich den Kopf über zuviele technische Einzelheiten zerbrechen zu müssen.

Zuerst wollen wir uns einmal auf die Sprite-Bewegungen konzentrieren, aber die gleichen Prinzipien können auch auf die Animation von BOBs oder Schirmen angewandt werden.

Zu Beginn müssen Sie einige Beispiel-Sprites in den Speicher laden. Sie finden sie auf der AMOS-Datendiskette im **Sprites**-Unterverzeichnis. Sie erhalten ein Verzeichnis aller Sprite-Dateiarten, indem Sie folgende Anweisung über das Direktfenster eingeben:

```
Dir "AMOS_DATA:"
```

Eine Sprite-Datei können Sie dann zum Beispiel folgendermaßen laden:

```
Load "AMOS_DATA:Sprites/octopus.abk"
```

Das Bewegen eines Objektes

Wie Sie es von einer speziellen Programmiersprache zur Animation sicher erwarten, können Sie Ihre Objekte mit AMAL auf viele verschiedene Arten bewegen. Am einfachsten ist hier der Einsatz des Move-Befehls.

Move *(Bewegt ein Objekt)*

M w,h,n

Mit dem M-Befehl bewegen Sie das Objekt *w* Einheiten nach rechts und *h* Einheiten nach unten. Dabei werden genau *n* Bewegungsschritte eingesetzt. Wenn die Koordinaten Ihres Objektes vorher (X,Y) lauteten, dann wird das Objekt somit zu den Koordinaten X+W,Y+H bewegt.

Angenommen ein Sprite befindet sich bei den Koordinaten 100,100. Die Anweisung M 100,100,100 bewegt es dann zu den Koordinaten 200,200. Die Geschwindigkeit dieser Bewegung hängt nun davon ab, in wievielen Einzelschritten diese Bewegung ausgeführt wird. Geben Sie für n einen großen Wert ein, dann ist die einzelne Sprite-Bewegung sehr klein und das Sprite bewegt sich sehr langsam. Entsprechend führt ein kleiner Wert für *n* zu großen Einzelschritten und das Sprite springt mit großer Geschwindigkeit über den Bildschirm. Hier sind nun einige Beispiele für den Move-Befehl:

```
Rem Hier wird ein Tintenfisch-Sprite mit AMAL auf dem Bildschirm nach unten bewegt
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,300,0,1
Amal 8,"M 0,250,50" : Amal On 8 : Wait Key

Rem Diese Version schiebt den Tintenfisch über den Bildschirm
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 9,150,150,1
Amal 9,"M 300,0,50" : Amal On 9 : Wait Key

Rem Bewegt den Tintenfisch auf dem Bildschirm rauf und runter
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 10,150,150,1
Amal 10,"M 300,-100,50" : Amal On 10 : Wait Key
```

```
Rem Demonstriert mehrere Move-Befehle
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
M$="Move 300,0,50; Move -300,0,50"
Sprite 11,150,150,1
Amal 11,A$ : Amal On 11 : Wait Key
```

Beachten Sie bitte, wie wir M in dem oben aufgeführten Programm zu Move ausgedehnt haben. Da die Buchstaben "ove" klein geschrieben sind, werden sie vom AMAL-System ignoriert.

Auf den ersten Blick ist Move eine zwar leistungsstarke, aber nicht besonders aufregende kleine Anweisung. Sie eignet sich zum Beispiel gut zum Bewegen von Geschossen, aber sonst ist sie doch etwas langweilig.

Wenn Sie das glauben, dann irren Sie sich aber ganz gewaltig. Die Parameter in der Move-Anweisung beschränken sich nämlich nicht nur auf einfache Zahlen. Sie können auch komplexe arithmetische Ausdrücke verwenden, durch die Sie eine Vielzahl von nützlichen AMAL-Funktionen ausdrücken können. Hier ist ein Beispiel dafür:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette : Sprite 12,150,150,1
Amal 12,"Move XM-X, YM-Y,32"
Amal On 12 : Wait Key
```

Diese Anweisung bewegt das berechnete Sprite mit der Nummer 12 glatt und fließend zur aktuellen Mausposition. X und Y geben die Koordination Ihres Sprites an. XM und YM sind Funktionen, die die aktuellen Mauskoordinaten ausweisen.

Diesen Effekt können Sie in Spielen wie Pac-Man ausnutzen, damit Ihre Objekte dem Männchen, das den Spieler darstellt, *hinterherjagen*. In **2** finden Sie eine Demonstration dieser Prozedur.

Mit dem Move-Befehl können Sie sogar den gesamten Bildschirm animieren. Hier ist ein einfaches Beispiel:

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.Iff",1
Channel 1 To Screen Display 1 : Rem Weist AMAL-Programm 1 dem Schirm 1 zu
Amal 1,"Move 0,-200,50 ; Move 0,200,50"
Amal On 1 : Direct
```

CHANNEL weist ein AMOS-Programm einem bestimmten Objekt zu. Wir werden diesen Befehl etwas später ausführlicher besprechen, aber sein Grundformat lautet folgendermaßen:

CHANNEL p To Objekt n

p ist dabei die Nummer Ihres AMAL-Programms. Die zulässigen Werten liegen hier im Bereich von 0 bis 63, obwohl nur die ersten 16 dieser Programme durch die Verwendung von Interrupts ausgeführt werden können.

Objekt gibt die Nummer das Objektes an, das Sie animieren wollen. Dieses Objekt muß danach durch die Anweisungen SPRITE, BOB oder SCREEN OPEN definiert werden. Hier sind einige Beispiele:

```
Channel 2 To Bob 1 : Rem Animiert Bob 1 durch AMAL-Programm Nummer 2
Channel 3 To Sprite 8 : Rem Weist Kanal drei einem berechneten Sprite zu
Channel 4 To Screen Display 0 : Rem Bewegt Default-Schirm mit AMAL
Channel 5 To Screen Offset 0 : Rem Verändert Schirm-Offset in AMAL
```

Animation

Anim (*Animiert ein Objekt*)

A n,(Bild,Verzögerung)(Bild,Verzögerung)...

Durch die Anim-Anweisung durchläuft das Objekt eine Reihe von Bildern. So wird ein glatter Animationseffekt hervorgerufen. n gibt dabei an, wie oft dieser Animationszyklus wiederholt werden soll. Wenn Sie für diesen Parameter den Wert Null einsetzen, dann wird die Animation ununterbrochen durchgeführt.

Bild gibt die Nummer des Bildes an, das für die einzelnen Rahmen Ihrer Animation eingesetzt wird. *Verzögerung* legt die Zeitspanne fest, die dieses Bild jeweils auf dem Bildschirm angezeigt wird. Sie wird in Fünfzigstelsekunden gemessen. Dazu folgende Beispiele:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,260,100,1
Amal 8,"A 0,(1,2)(2,2)(3,2)(4,2)"
Amal On 8 : Direct

Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,50,11
M$="Anim 12, (1,4)(2,4)(3,4)(4,4)(5,4)(6,4) ; "
M$=M$+"Move 300,150,150 ; Move -300,-150,75"
Amal 9,M$
Amal On 9
Direct
```

Im zweiten Beispiel wird eine Sprite-Bewegung mit einer Animation verbunden. Beachten Sie, wie wir die Befehle durch den Strichpunkt ";" voneinander abgetrennt haben. So wird sichergestellt, daß die beiden Operationen völlig unabhängig voneinander sind. Nachdem die Animationssequenz definiert ist, springt AMAL sofort zur nächsten Anweisung und die Animation beginnt.

Dabei sollten Sie nicht vergessen, daß Anim nur in Verbindung mit Sprites oder BOBs funktioniert. Mit diesem Befehl können Sie also nicht den gesamten Bildschirm animieren.

Einfache Schleifen

Jump *(Umleiten eines AMAL-Programmes)*

J Label

Mit Jump können Sie auf einfache Art von einem Teil eines AMAL-Programmes zu einem anderen springen. *Label*, die Sprungmarke, stellt dabei das Ziel Ihres Sprunges dar, und muß zuvor an einer anderen Stelle Ihres aktuellen Programmes definiert worden sein.

Alle AMAL-Sprungmarken werden durch einen Großbuchstaben gefolgt vom einem Doppelpunkt definiert. Wie die AMAL-Anweisungen können Sie auch die Sprungmarken durch Kleinbuchstaben erweitern, damit sie leichter zu lesen sind. Hier sind einige Beispiele:

S:
Swoop:
Label:

Vergessen Sie nicht, daß jede Sprungmarke durch einen **einzigen** Buchstaben definiert wird. Folglich beziehen sich S und Swoop auf **dieselbe** Sprungmarke! Wenn Sie versuchen, zwei Sprungmarken zu definieren, die mit demselben Buchstaben beginnen, erhalten Sie die Fehlermeldung *Sprungmarke in Animationskette bereits definiert.*

Jedes AMAL-Programm kann über einen eigenen Satz von Sprungmarken verfügen. Es ist also völlig zulässig, identische Sprungmarken in den verschiedenen Programmen einzusetzen. Dazu folgendes Beispiel:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Rem Stellt sieben berechnete Sprites auf den Bildschirm
For S=8 To 20 Step 2
   Sprite S,200,(S-7)*13+40,1
Next S
Rem Erzeugt sieben AMAL-Programme
For S=1 To 7
   Channel S to Sprite 6+(S*2)
   M$="Anim 0,(1,2)(2,2)(3,2)(4,2) ; Label: Move "+Str$(S*2)+",0,7;"
   M$=M$+"Move -"+Str$((6-2)*2)+",0,7 ; Jump Label"
   Amal S,M$
Next S
Rem Ok, und jetzt wird alles animiert!
Amal On : Direct
```

Im nächsten Beispiel wird ein Sprite wiederholt zur aktuellen Mausposition bewegt.

```
Load "AMOS_DATA:Sprites/octopus.abk"
Get Sprite Palette
Sprite 8,150,150,1
Amal 8,"R: Move XM-X,YM-Y,8 ; Pause; Jump R"
Amal On 8
```

Da die AMAL-Befehle durch Interrupts ausgeführt werden, könnten Endlosschleifen verheerende Auswirkungen haben. Deshalb wird die Anzahl der Sprünge in Ihrem Programm automatisch gezählt. Wenn diese Anzahl 10 überschreitet, dann ignoriert das AMAL-System alle weiteren Sprünge.

Achtung: Wenn Sie sich aber völlig auf dieses System verlassen, und in Ihren Programmen Endlosschleifen zulassen, verschwenden Sie einen Großteil der Rechnerleistung des Amiga. In der Praxis ist es wesentlich wirksamer, sich auf einen Sprung pro VBL zu beschränken. Das erreichen Sie, indem Sie vor jeder Jump-Anweisung einen PAUSE-Befehl in Ihr Programm einfügen. Weitere Einzelheiten dazu finden Sie unter PAUSE.

Variablen und Ausdrücke

Let *(Weist einem Register einen Wert zu)*

L register=expression

Die L-Anweisung weist einem AMAL-Register einen Wert zu. Dies geht so ähnlich wie im normalen Basic vor sich, jedoch werden hier alle Ausdrücke streng von links nach rechts berechnet.

Register sind Ganzzahl-Variablen, die die Zwischenergebnisse in Ihren AMAL-Programmen beinhalten. Die zulässigen Werte liegen zwischen -32768 und +32767. Es gibt drei Grundtypen von Registern:

Interne Register

Jedes AMAL-Programm verfügt über einen eigenen Satz von 10 *internen* Registern. Die Namen dieser Register beginnen mit dem Buchstaben R gefolgt von einer der Zahlen von 0 bis 9 (R0 -R9).

Interne Register entsprechen den in einer AMOS Basic-Prozedur definierten lokalen Variablen.

Externe Register

Externe Register sind etwas anders, denn sie behalten ihren Wert in verschiedenen AMAL-Programmen bei. So können Sie durch diese Register zum Beispiel zwischen verschiedenen AMAL-Routinen Informationen übertragen. AMAL bietet Ihnen bis zu 26 externe Register mit den Namen RA bis RZ.

Über die (später beschriebene) AMREG-Funktion erhalten Sie direkt aus Ihrem Basic-Programm heraus Zugang zum Inhalt jedes internen und externen Registers.

Spezialregister

Spezialregister stellen einen Satz von drei Werten dar, die den Status Ihres Objektes bestimmen.

X,Y enthalten die Koordinaten Ihres Objektes. Durch Verändern dieser Register können Sie Ihre Objekte auf dem Bildschirm bewegen. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Sprites/Frog_Sprites.abk" : Channel 1 To Bob 1
Flash Off : Get Sprite palette : Bob 1,0,0,1
Amal 1,"Loop: Let X=X+1 ; Let Y=Y+1; Pause; Jump Loop"
Amal On 1 : Direct
```

A speichert die Nummer des Bildes, das von einem Sprite oder BOB dargestellt wird. Sie können diesen Wert verändern, um Ihre eigenen Animationssequenzen folgendermaßen zu generieren:

```
Load "AMOS_DATA:Sprites/Frog_Sprites.abk" : Get Sprite Palette : Flash Off
Channel 2 To Bob 1 : Bob 1,300,100,1
M$="Loop: Let A=A+1 ; "
M$=M$+"For R0=1 To 5 ; Next R0 ; Jump Loop"
Amal 2,M$
Amal On 2 : Direct
```

Die Schleife *For To Next* wird weiter unten noch genauer erklärt. Hier wird sie eingesetzt, um jede Veränderung des BOB-1-Bildes zu verlangsamen. Wenn das *Next* der Schleife ausgeführt worden ist, fährt AMAL erst nach dem nächsten Vertical Blank fort. Beachten Sie bitte auch die Verwendung des Strichpunktes ";", durch den die AMAL-Anweisungen voneinander abgetrennt werden. Sie könnten dafür natürlich genausogut einen Leerschritt einsetzen.

Operatoren

AMAL-Ausdrücke können alle normalen arithmetischen Operationen außer MOD umfassen. Sie können in Ihren Berechnungen auch die folgenden logischen Operationen einsetzen:

&	Logisches UND (AND)
I	Logisches ODER (OR)

Beachten Sie bitte, daß Sie die Reihenfolge der Berechnung nicht durch das Einsetzen von Klammern () verändern können. Dies würde Ihre Berechnungen erheblich verlangsamen und daher die für das Interrupt zulässige Zeitspanne verringern. Hier sind noch einige weitere Beispiele, die Sie eingeben können:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Hide
Get Sprite Palette
Sprite 8,X MOUSE,Y MOUSE,1
Amal 8,"Loop: Let X=XM ; Let Y=YM ; Pause ; Jump Loop"
Amal On 8
```

```
Load "AMOS_DATA:Sprites/octopus.abk" : Hide
Get Sprite Palette
Sprite 8,X MOUSE,Y MOUSE,1
Amal 8,"Anim 0,(1,4)(2,4),(3,4)(4,4) ; Loop: Let X=XM ; Let Y=YM ; Pause ; Jump Loop"
Amal On
```

Die vorstehenden Beispiele simulieren effektiv den Befehl CHANGE MOUSE. Die Leistung dieses Systems ist jedoch wesentlich stärker, denn Sie können hier mit genau der gleichen Technik BOBs, berechnete Sprites und sogar Bildschirme ganz leicht bewegen.

Das Treffen von Entscheidungen

If *(Verzweigung in einer AMAL-Zeichenkette)*

If Test Jump L

Mit dieser Anweisung können Sie in Ihrem AMAL-Programmen einfache Abfragen durchführen. Wenn der Ausdruck *test* gleich -1 (richtig) ist, dann springt das Programm zur Sprungmarke *3*, ansonsten geht AMAL unverzüglich zur nächsten Anweisung weiter. Anders als beim gleichwertigen Basic-Befehl sind Sie hier jedoch nach der Abfrage auf einen einzigen Sprung beschränkt.

Normalerweise ist es üblich, diese Anweisung mit kleingeschriebenen Befehlen wie "then" oder "else" auszupolstern. Auf diese Art wird die Wirkung der Anweisung wesentlich deutlicher. Hier ist ein Beispiel dazu:

```
If X>100 then Jump Label else Let X=X+1
```

Test kann jeder beliebige logische Ausdruck sein, und zum Beispiel:

<>	Ungleich
<	Kleiner als
>	Größer als
=	Gleich

enthalten. Hier ist ein Beispiel dazu:

```
Load "AMOS_DATA:Sprites/octopus.abk"
```

```
Get Sprite Palette
Sprite 8,130,50,1
C$="Main: If XM>100 Jump Test; "
C$=C$+" Let X=XM"
C$=C$+" Test: If YM>100 Jump Main: "
C$=C$+" Let Y=YM Jump Main"
Amal 8,C$ : Amal On : Direct
```

In **4** können Sie ein umfassenderes Beispiel finden, in dem Sie die Position eines Sprites mit dem Joystick steuern können. Das ist aber eigentlich noch eine Rohform, die mit Hilfe des Befehls AUTOTEST drastisch beschleunigt werden kann. Siehe dazu den Befehl AUTOTEST.

Achtung! Versuchen Sie nie, mehrere Abfragen durch "&" oder "I" in einem einzigen AMAL-Ausdruck miteinander zu verbinden. Da alle Ausdrücke von links nach rechts berechnet werden, würden Sie dann eine Fehlermeldung erhalten. Nehmen Sie zum Beispiel den Ausdruck: X> 100IY> 100. Hier wollen Sie überprüfen, ob X> 100 ODER Y> 100 ist. In der Praxis wird dieser Ausdruck jedoch in der folgenden Reihenfolge ausgewertet:

X> 100	Kann RICHTIG (TRUE) oder FALSCH (FALSE) sein
IY	Verknüpfung des Ergebnisses mit Y durch OR (ODER)
>100	Überprüft ob (Y>100IY)>100)

Das Ergebnis des obengenannten Ausdrucks weist offensichtlich keinen Bezug zum erwarteten Wert auf. Technisch orientierte Anwender können dieses Problem durch den Einsatz der Booleschen Algebra umgehen. Zuerst weisen Sie jeder Abfrage folgendermaßen ein bestimmtes AMAL-Register zu:

```
Let R0=X>100; Let R1=Y>100
```

Dann verbinden Sie diese Abfragen mit "&" oder "I" zu einem einzigen Ausdruck, denn Sie nun direkt in Ihre If-Anweisung einsetzen.

```
If R0 I R1 then Jump L ...
```

Das sieht vielleicht etwas komisch aus, aber in der Praxis funktioniert es wunderbar.

For To Next *(Schleife in AMAL)*

For Reg=start To Ende
: :
Next Reg

Durch diese Anweisung wird eine normale FOR...NEXT Schleife ausgeführt. Sie entspricht der gleichlautenden Basic-Anweisung fast vollkommen. Sie können diese Schleifen in Ihren Programmen dazu benutzen, Objekte in komplexen visuellen Mustern zu bewegen. Reg kann jedes normale AMAL-Register (R0-R9 oder RA-RZ) bezeichnen. Sie können die Spezialregister zu diesem Zweck jedoch nicht verwenden.

Wie beim Basic muß das auf Next folgende Register mit dem Zähler, den Sie in For bestimmt haben, übereinstimmen, sonst erhalten Sie einen AMAL-Syntaxfehler. Beachten Sie auch, daß die Schrittgröße immer auf eins gestellt ist. Sie können jedoch darüber hinaus eine beliebige Anzahl von Schleifen ineinander schachteln.

Jeder Animationskanal führt pro VBL immer nur eine Schleife aus. So wird die Wirkung Ihrer Schleifen mit der Bildschirmanzeige synchronisiert und es besteht keine Notwendigkeit, einen ausdrücklichen Pause-Befehl vor jedem Next einzufügen.

Wie Sie eine Angriffswelle für ein Spiel generieren

Mit diesen Schleifen kann man ziemlich komplexe Bewegungsabläufe hervorrufen. Die einfachste Art der Bewegung ist in einer geraden Linie. Das können Sie durch eine einzige For..Next Schleife folgendermaßen hervorrufen:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,60,1
C$="For R0=1 To 320 ; Let X=X+1 ; Next R0" : Rem Bewegt Sprite von links nach rechts
Amal 8,C$ : Amal On 8 : Direct
```

Sie können dieses Programm jetzt erweitern, um das Objekt auf dem Bildschirm hin- und herzuziehen.

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,60,1
C$="Loop: For R0=1 To 320 ; Let X=X+1 ; Next R0" : Rem Schiebt Sprite nach vorne
C$=C$+"For R0=1 To 320 ; Let X=X-1 ; Next R0 ; Jump Loop" : Rem Zieht Sprite nach hinten
Amal 8,C$ : Amal On 8 : Direct
```

Die erste Schleife bewegt das Objekt von links nach rechts, und die zweite von rechts nach links.

Bis jetzt war das Muster nur auf horizontale Bewegungen beschränkt. Um aber jetzt eine realistische Angriffswelle zu erzeugen, muß man in diese Bewegung auch eine vertikale Komponente miteinbeziehen. Das erreichen Sie, indem Sie in Ihr Programm noch eine weitere Schleife einbauen.

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,60,1
C$="For R1=0 To 10 ;"
C$=C$+"For R0=1 To 320 ; Let X=X+1 ; Next R0" : Rem Geht nach vorne
C$=C$+"Let Y=Y+8;": Rem Bewegt Sprite auf dem Bildschirm nach unten
C$=C$+"For R0=1 To 320 ; Let X=X-1 ; Next R0 ;" : Rem Geht nach hinten
C$=C$+"Let Y=Y+8; Next R1": Rem Bewegt Sprite nach unten
Amal 8,C$ : Amal On 8 : Direct
```

Mit dem obengenannten Programm erzeugen Sie ein glattes, aber ziemlich einfaches Angriffsmuster. In **BEISPIEL 14.1** im MANUAL-Unterverzeichnis finden Sie eine weitere Demonstration.

Das Erfassen eines komplexen Bewegungsablaufs

PLay

PLay Pfad

Wenn Sie sich die glatten Angriffswellen in einem der beliebten Ballerspiele angesehen haben, und dabei dachten, daß Sie selbst so was nie zustande bringen würden, dann haben wir jetzt eine Überraschung für Sie bereit. Mit dem AMAL- Befehl Play können Sie Ihre Objekte mit praktisch jeder beliebigen Bewegungsfolge animieren. Das funktioniert, indem eine vorher definierte, in der AMAL Speicherbank gespeicherter Bewegungsfolge abläuft.

Diese Bewegungsabläufe werden mit Hilfe des AMAL-Zusatzprogrammes auf der AMOS-Programmdiskette erzeugt. Dieses Programm erfaßt einfach eine Folge von Mausbewegungen und speichert sie direkt in die AMAL Speicherbank. Wenn Sie Ihre Muster einmal auf diese Art definiert haben, können Sie sie ohne jede Anstrengung ganz einfach jedem beliebigen Objekt auf dem Bildschirm zuweisen, dann werden Ihre ursprünglichen Muster ganz genau nachvollzogen. Sie können aus Ihrem AMOS Basic-Programm heraus jederzeit Geschwindigkeit und Richtung der Bewegung verändern.

Wenn AMAL zum ersten Mal auf einen Play-Befehl stößt, überprüft es die AMAL-Bank, um die gespeicherte Bewegung, die Sie durch den Parameter *Pfad* definiert haben, zu finden. *Pfad* stellt einfach eine Nummer dar, die zwischen eins und der höchstmöglichen Anzahl der Muster in der Bank liegt. Sollte in dieser Phase ein Problem auftreten, bricht AMAL die Ausführung der Play-Anweisung völlig ab und springt zur nächsten Anweisung in Ihrer Animationskette.

Nachdem das Muster initialisiert wurde, wird das Tempo der Bewegung in das Register R0 geladen. Dies bestimmt die Zeitspanne zwischen jedem einzelnen Bewegungsschritt. Alle Zeitmessungen erfolgen in Einheiten von Fünfzigstelsekunden. Wenn Sie dieses Register in Ihrem AMAL-Programm verändern, können Sie Ihre Objekte entsprechend beschleunigen oder verlangsamen.

Beachten Sie, daß jeder Bewegungsschritt zu den gegenwärtigen Koordinaten Ihres Objektes **addiert** wird. Wenn also ein Objekt in der Folge mit den Sprite- oder BOB-

Anweisungen bewegt wird, so wird es sich von der neuen Bildschirmposition aus völlig unbeeinflußt weiterbewegen. Deshalb können Sie mit einer einzigen Bewegungsfolge eine Vielzahl verschiedener Objekte auf dem Bildschirm animieren.

Register R1 enthält jetzt ein Flag, das die Richtung Ihrer Bewegungen bestimmt. Es gibt hier drei mögliche Situationen:

R1>0 Vorwärts

Ein Wert von eins für R1 bestimmt, daß der Bewegungsablauf von Anfang bis Ende genau in der Reihenfolge, in der er erzeugt wurde, durchgeführt werden soll (Default).

R1=0 Rückwärts

Viele Animationssequenzen erfordern es, daß Ihre Objekte in einem komplexen Muster auf dem Bildschirm hin- und herlaufen. Um die Bewegungsrichtung zu ändern, laden Sie einfach R1 mit 0. Jetzt dreht sich Ihr Objekt um und führt Ihre ursprünglichen Bewegungsschritte rückwärts aus.

R1=-1 Exit

Wenn von Ihrem AMOS-Programm eine Kollision festgestellt wurde, müssen Sie Ihr Objekt sofort anhalten und einen explosionsartigen Effekt hervorrufen. Das erreichen Sie, indem Sie R1 auf den Wert -1 setzen. Jetzt bricht AMAL die Play- Anweisung ab und springt direkt zur nächsten Anweisung in Ihrer Animationssequenz.

Das Raffinierte bei diesen Registern ist, daß sie direkt aus AMOS-Basic heraus verändert werden können. Auf diese Art können Sie Ihre Bewegungsabläufe direkt aus Ihrem Hauptprogramm heraus steuern. Und eine spezielle AMPLAY-Anweisung macht Ihnen die Sache noch leichter.

Der PLay-Befehl eignet sich ideal zur Steuerung der Aliens in einem Baller- oder Abenteuerspiel. Er stellt in der Tat eigentlich allein für sich betrachtet die stärkste der AMAL-Anweisungen dar.

AMAL *(Ruft ein AMAL-Programm auf)*

AMAL n,a$
AMAL n,p
AMAL n,a$ to Adresse

Der AMAL-Befehl weist einem AMAL-Programm einen Animationskanal zu. Dieses Programm wird entweder der Zeichenkette *a$* oder direkt der AMAL-Bank entnommen.

Die erste Version dieser Anweisung lädt Ihr Programm aus der Zeichenkette a$ und weist es dem Kanal *n zu. a$* kann jede beliebige Reihe von AMAL-Anweisungen enthalten. Alternativ dazu können Sie Ihr Programm auch aus einer Speicherbank laden, die mit Sie mit dem AMAL-Zusatzprogramm erzeugt haben. Dann gibt *p* die Nummer eines in der Bank mit der Nummer 4 gespeicherten AMAL-Programmes an.

n ist die Nummer eines Animationskanals und liegt zwischen 0 und 63. Jeder AMOS-Kanal kann entweder einem BOB, Sprite oder Schirm zugewiesen werden.

Nur die ersten 16 AMAL-Programme können durch Interrupts ausgeführt werden. Diese Begrenzung können Sie umgehen, indem Sie Ihre Programm direkt aus dem Basic mit dem Befehl SYNCHRO aufrufen.

Die letzte Version der AMAL-Anweisung ist für die fortgeschrittenen Anwender gedacht. Anstatt ein echtes Objekt zu bewegen, kopiert sie den Inhalt der Register X,Y und A in einen bestimmten Speicherbereich. Sie können diese Information jetzt direkt in Ihren eigenen Basic-Routinen einsetzen. Deshalb können Sie mit dem AMAL-System auch alles Mögliche von BLOCK bis zu einem Zeichen animieren. Das Format lautet:

AMAL n, a$ To Adresse

Adresse muß GERADE sein und auf einen sicheren Bereich im Speicher weisen, vorzugsweise in einer AMOS Zeichenkette oder Speicherbank. Jedesmal wenn Ihr AMAL-Programm ausgeführt wird (50mal pro Sekunde), werden die folgenden Werte in diesen Speicherbereich geschrieben:

Speicherzelle	Wirkung
Adresse	Bit 0 wird auf 1 gestellt, wenn X verändert wurde. Bit 1 gibt an, daß Y verändert wurde. Bit 2 wird gesetzt, wenn das Bild (A) seit dem letzten Interrupt verändert wurde.
Adresse+2	Ist ein **Wort**, das den neuesten Wert von X enthält.
Adresse+4	Enthält den aktuellen Wert von Y.
Adresse+6	Speichert den Wert von A.

Durch ein einfaches DEEK erhalten Sie aus Ihrem Programm Zugang zu diesen Werten.
Achtung: Diese Option überlagert alle vorherigen CHANNEL-Zuweisungen.

AMAL-Befehle

Hier ist eine vollständige Liste der verfügbaren AMAL-Befehle:

Move

Dieser Anweisung bewegt ein Objekt ganz glatt von einer Position in eine andere. Die Syntax lautet folgendermaßen:

Move DeltaX, DeltaY, Schritte

DeltaX gibt den Abstand der horizontalen Bewegung an. Positive Zahlen reflektieren eine Bewegung von links nach rechts, negative Werte eine Bewegung von rechts nach links.

DeltaY bestimmt die vertikale Verschiebung. Ist DeltaY positiv, so bewegt sich Ihr Objekt auf dem Bildschirm nach unten, ansonsten bewegt es sich nach oben.

n stellt die Anzahl der Schritte dar, in denen die Bewegung durchgeführt wird. Die glattesten Bewegungen lassen sich erzielen, wenn sowohl *DeltaX* wie auch *DeltaY* Vielfache von *n* sind.

A (Anim)

Anim Zyklen,(Bild,Verzögerung)(Bild,Verzögerung)...

Die Anim-Anweisung weist eine Folge von Bildern entweder einem Sprite oder einem Blitter-Objekt zu und erzeugt so einen realistischen Animationseffekt.

Zyklen gibt an, wie oft die Animation wiederholt werden soll. Wenn Sie diesen Parameter auf Null setzen, läuft die Animation endlos weiter ab. *Bild* wählt die Bildnummer für jeden einzelnen Rahmen Ihrer Animation. *Verzögerung* legt die Zeitspanne (in Fünfzigstelsekunden) fest, in der das Bild jeweils angezeigt wird.

Nachdem der Anim-Befehl initialisiert wurde, springt AMAL automatisch zur nächsten Anweisung. So können Sie im selben AMAL-Programm Bewegung und Animation miteinander verbinden.

Let

Let Reg=Exp

Dieser Befehl weist einem AMAL-Register einen Wert zu. Reg ist der Name des AMAL-Registers, das verändert werden soll. Es stehen Ihnen 10 interne Register (R0 bis R9) und des weiteren 26 externe Register (RA bis RZ) zur Verfügung. Sie können über die Spezialregister X, Y und A auch Position und Art Ihres Objektes direkt ändern.

Exp ist ein arithmetischer Standardausdruck und wird zur Ermittlung des Endergebnisses von links nach rechts berechnet.

Die meisten normalen Operatoren werden unterstützt, einschließlich +,-,* und /. Sie dürfen die Reihenfolge der Berechnung jedoch nicht durch Klammern () verändern.

Jump

Jump L

Mit dem Jump-Befehl können Sie vom gegenwärtigen Standpunkt in Ihrem AMAL-Programm zur Sprungmarke *L* springen. *L* ist die Bezeichnung einer Sprungmarke, die Sie zuvor in Ihrer AMAL-Zeichenkette definiert haben. Sprungmarken bestehen aus einem einzigen Großbuchstaben und werden wie im normalen Basic mit einem Doppelpunkt ":" gekennzeichnet.

If

If Exp Jump L

Die If-Anweisung ermöglicht es Ihnen, aufgrund eines Testergebnisses von einem Teil eines AMAL-Programmes zu einem anderen zu springen. Exp stellt einen logischen Ausdruck im Standardformat dar.

Wenn das Ergebnis von Exp RICHTIG (TRUE) lautet, dann springt das Programm zur Sprungmarke *3*, sonst wird sofort die nächste Anweisung ausgeführt.

Es gibt von diesem Befehl zwei Ausführungen, die von der AUTOTEST-Funktion eingesetzt werden:

If Exp Direct L	(Wählt den Teil des Programmes, der nach demAutotest ausgeführt werden soll)
If Exp eXit	(Verläßt Autotest)

Weitere Einzelheiten siehe unter AUTOTEST.

For To Next

For Reg=Start To Ende...Next Reg

Dies ist die direkte Umsetzung der FOR...NEXT Schleifen aus dem Basic. *Reg* kann dabei jedes beliebige interne oder externe AMAL-Register darstellen. Wie üblich können die Schleifen auch geschachtelt werden, die Schrittgröße ist jedoch immer auf eins gesetzt.

Beachten Sie, daß AMAL automatisch auf den nächsten Vertical Blank wartet, bevor es mit Next an den Anfang der Schleife zurückspringt. Da die Bewegungen des Objektes erst zu sehen sind, nachdem der Schirm nach dem VBL aktualisiert wurde, würden schnellere Schleifen nur kostbare Rechnerzeit ohne sichtbare Wirkung verschwenden. Also werden Ihre For...Next Schleifen stets automatisch mit der Aktualisierung des Bildschirms synchronisiert und erzeugen so die glattesten und fließendsten Bewegungen.

PLay

PLay Pfad

Der PL-Befehl animiert Ihre Objekte durch eine Reihe von Bewegungen, die in der AMAL-Bank gespeichert wurden. Diese Abläufe werden direkt mit der Maus über das leistungsstarke AMAL-Zusatzprogramm eingegeben. Deshalb ist die Art der Bewegungsabläufe, die Sie mit diesem System erzeugen können, praktisch unbegrenzt.

Pfad ist die Nummer des Bewegungsablaufs, der zuvor in der AMAL-Bank gespeichert wurde. Wenn dieser Ablauf nicht existiert, übergeht AMAL die PL-

Anweisung und springt sofort zum nächsten Befehl in Ihrer Animationssequenz.

Alle Bewegungen werden im Verhältnis zur gegenwärtigen Position Ihrer Objekte ausgeführt. Deshalb können Sie mit einer einzigen Pfad-Definition zum Beispiel eine ganze Angriffswelle bewegen. Sie können außerdem ein Objekt direkt aus dem Basic heraus bewegen, ohne daß das die geringste Auswirkung auf die gespeicherte Bewegung hat. Der Status der gegenwärtigen Bewegung wird durch zwei AMAL-Register gesteuert.

R0 enthält das Tempo Ihrer Bewegung. Wenn Sie diesen Wert erhöhen, wird die Bewegung des Objektes auf dem Bildschirm beschleunigt.

R1 enthält die Richtung der Bewegung. Es gibt drei mögliche Alternativen.

R1>0 Durchläuft die Bewegungsfolge in der ursprünglichen Reihenfolge.

R1=0 Führt Ihre Bewegungsschritte rückwärts aus.

R1=-1 Bringt die Bewegungsfolge völlig zum Stillstand und schreitet zur nächsten AMAL-Anweisung fort.

Sie können den Inhalt dieser Register jederzeit aus Ihrem Basic-Programm heraus entweder über den AMREG- oder den speziellen AMPLAY-Befehl verändern.

Eine weitere Erklärung dieser Anweisung finden Sie im AMAL-Tutorial am Anfang dieses Kapitels. Siehe auch **BEISPIEL 14.2** im MANUAL-Unterverzeichnis.

Achtung: Sie müssen Ihre AMAL-Anweisungen stets unbedingt durch Semikolon voneinander abtrennen. Die folgende Zeichenkette ruft zum Beispiel die Fehlermeldung *AMAL Bank nicht reserviert* nur deshalb hervor, weil hier die Unterteilung der Anweisung fehlt.

```
A$="Pause Let R0=1"
```

Die korrekte Syntax für diese Anweisung lautet folgendermaßen:

```
A$="Pause ; Let R0=1"
```

End

End

End beendet das gesamte AMAL-Programm und schaltet die Autotest-Funktion ab, wenn sie zuvor definiert worden wurde.

Pause

Pause

Pause hält zeitweise die Ausführung Ihres AMAL-Programms an und wartet auf den nächsten Vertical Blank. Nach dem VBL wird das Programm von der nächsten Anweisung an automatisch wieder aufgenommen.

Pause wird oft vor einem Jump-Befehl verwendet, damit sichergestellt ist, daß die Anzahl der Sprünge unter der Höchstanzahl von 10 pro VBL liegt. Dadurch wird wertvolle Rechnerzeit für Ihre Basic-Programme frei, und das kann eine durchschlagende Wirkung auf ihre Gesamtgeschwindigkeit haben. Deshalb sollten Sie sich angewöhnen, Ihren Jump-Befehlen immer eine Pause-Anweisung voranzustellen, da dies viel effizienter ist.

AUtotest

AU (List of tests)

Die AMAL Autotest-Funktion wurde entwickelt, um eine schnelle Interaktion zwischen AMAL und dem Anwender zu ermöglichen. Es wird am Anfang des AMAL-Programms eine besondere Abfrage hinzugefügt, die bei jedem Vertical Blank durchgeführt wird, bevor der Rest des AMAL-Programmes ausgeführt wird. Nähere Einzelheiten dazu finden Sie unter dem Autotest-System.

eXit

eXit

Steigt aus einem Autotest aus und kehrt zum aktuellen AMAL-Programm zurück.

Wait

Wait

Wait unterbricht Ihr AMAL-Programm und führt nur den Autotest durch.

On

On

ON aktiviert das Hauptprogramm nach dem Wait-Befehl wieder.

Direct

Direct

Bestimmt den Teil des Hauptprogramms, das nach einem Autotest durchgeführt werden soll.

Die AMAL-Funktionen

=XM *(Gibt die X-Koordinate der Maus an)*

Diese Funktion stimmt genau mit der X MOUSE Funktion in AMOS-Basic überein. Sie weist die X-Koordinate des Maus-Cursors in Hardware-Koordinaten aus.

=YM *(Gibt die Y-Koordinate der Maus an)*

YM weist die Y-Koordinate des Mauszeigers in Hardware-Koordinaten aus.

=K1 *(Status der linken Maustaste)*

K1 gibt einen Wert von -1 (richtig) an, wenn die Maustaste gedrückt wurde, sonst erscheint 0 (falsch).

=K2 *(Status der rechten Maustaste)*

Gibt den Status der rechten Maustaste an. Wenn sie gedrückt wurde, zeigt K2 den Wert -1 (richtig).

=J0 *(Fragt rechten Joystick ab)*

Die J0-Funktion fragt den rechten Joystick ab und weist dann eine Bitmap aus, die den gegenwärtigen Status enthält. Weitere Einzelheiten dazu finden Sie unter JOY.

=J1 *(Fragt den linken Joystick ab)*

Diese Funktion fragt den linken Joystick ab und zeigt ein Bitmuster im Standardformat an.

=Z(n) *(Zufallszahl)*

Durch die Z-Funktion erhalten Sie eine Zufallszahl zwischen -32767 und 32768. Durch den Einsatz der Bitmaske *n* kann der Bereich, aus dem diese Zahl stammt, eingeschränkt werden.

Zwischen der Bitmaske *n* und der Zufallszahl wird eine logische UND-Operation durchgeführt, aus der dann das Endergebnis hervorgeht. Wenn Sie also für *n* einen Wert von 255 festlegen, dann liegen die ausgewiesenen Zahlen im Bereich zwischen 0 und 255.

Da diese Funktion zugunsten der Geschwindigkeit optimiert wurde, ist die ausgewiesene Zahl keine reine Zufallszahl. Wenn Sie ganz reine Zufallszahlen benötigen, dann sollten Sie Ihre Werte mit der RND-Funktion in Basic generieren und sie dann mit der AMREG-Funktion in ein externes AMAL-Register laden.

=XH *(Konvertiert eine Bildschirm-X-Koordinate in eine Hardware-Koordinate)*

=XH(s,x)

Diese Funktion konvertiert eine Bildschirmkoordinate in die entsprechende Hardware-Koordinate bezogen auf den Schirm mit der Nummer s.

=YH *(Konvertiert eine Bildschirm-Y-Koordinate in das Hardware-Format)*

=YH(s,y)

YH setzt eine Y-Koordinate vom Schirmformat in das entsprechende Hardware- Format für den Schirm *s* um.

=XS *(Konvertierung von Hardware- zu Bildschirmkoordinate)*

=XS(s,x)

Verändert die Hardware-Koordinate *x* in eine Grafikkoordinate mit Bezug zum Schirm *s*.

=YS *(Konvertierung von Hardware- zu Bildschirmkoordinate)*

=YS(s,y)

Wandelt Hardware-Koordinate *y* in die entsprechende Bildschirmkoordinate um.

=BC *(Kollisionsabfrage für BOBs)*

=Bob Col(n,s,e)

BC ist mit der entsprechenden BOB COL Anweisung in AMOS-Basic identisch. Diese Funktion überprüft das BOB mit der Nummer *n* nach Kollisionen mit den BOBs im

Bereich zwischen *s* und *e*.

Wird eine Kollision festgestellt, so gibt BC den Wert -1 (richtig) aus, sonst erscheint 0 (falsch). Diese Anweisung kann **nicht** in einem Interrupt durchgeführt werden. Sie ist also nur verfügbar, wenn Sie Ihre AMAL-Routinen direkt aus dem Basic mit der SYNCHRO-Anweisung abrufen.

=SC *(Sprite-Kollision)*

=Sprite Col(n,s,e)

Diese Anweisung stimmt genau mit der Funktion SPRITE COL in AMOS-Basic überein. Hier wird das Sprite n nach Kollisionen mit den Sprites s bis e abgefragt. Wenn die Abfrage erfolgreich ist, so wird der Wert -1 (richtig) ausgewiesen. Wie die obengenannte BC-Funktion kann auch sie nur in Verbindung mit der SYNCHRO- Anweisung eingesetzt werden.

=C *(Col)*

=C(n)

Gibt den Status des Objektes *n* nach dem Einsatz der SC- oder BC-Funktion an. Wenn ein Objekt kollidiert ist, wird ein Wert von -1 (richtig), ansonsten 0 (falsch) ausgewiesen.

=V *(Vumeter)*

=V(v)

Die VU-Funktion mißt einen der Tonkanäle und zeigt die Intensität der aktuellen Stimme an. Das ergibt dann eine Zahl zwischen 0-255. Sie können diese Information dazu verwenden, Ihre Objekte im Takt mit der Musik zu animieren. Sie finden in **BEISPIEL 14.3** ein Beispiel dafür. Siehe auch die VUMETER-Funktion in AMOS-Basic.

Das Steuern von AMAL aus dem Basic

AMAL ON/OFF *(Starten/Stoppen eines AMAL-Programmes)*

AMAL ON/OFF [n]

Nachdem Sie Ihr AMAL-Programm definiert haben, müssen Sie es mit dem Befehl AMAL ON ausführen. So wird Ihr AMAL-System aktiviert und Ihre Programme von der ersten Anweisung an ausgeführt.

AMAL ON aktiviert alle Ihre Programme. Über den Parameter n können Sie wahlweise jeweils nur eine Routine starten.

AMAL FREEZE *(Hält ein AMAL Programm zeitweise an)*

AMAL FREEZE[n]

Hält ein laufendes AMAL Programm zeitweise an. Alle Bewegungen werden eingefroren. Das Programm kann aber jederzeit wieder gestartet werden, mit einem einfachen Aufruf von AMAL ON. AMAL FREEZE sollte vor jedem grösseren Disk-Zugriff wie etwa Dir angewendet werden, da sonst ernste Programme mit dem Timing auftreten können.

AMAL OFF [n]

Stoppt die Ausführung aller - oder der durch den Parameter n bezeichneten - AMAL-Programme. Sie können Ihre Programme jederzeit einfach durch Aufrufen von AMAL ON wieder starten. Beachten Sie, daß Sie AMAL stets durch diese Anweisung anhalten sollten, bevor Sie zum Beispiel einen Befehl wie DIR ausführen, sonst können Probleme mit dem Timing zu visuellen Pannen führen.

=AMREG= *(Ermittelt den Wert eines externen AMAL-Registers)*

r=AMREG ([Kanal],n)
AMREG ([Kanal],n)=Ausdruck

Über die AMREG-Funktion erhalten Sie Zugang zum Inhalt der internen und externen AMAL-Register direkt aus Ihrem Basic-Programm heraus.

n stellt die Nummer des Registers dar. Die Werte können im Bereich zwischen 0 und 25 liegen, dabei stellt 0 das Register RA und der Wert 25 RZ dar.

Wenn Sie die Eingabe des Parameters Kanal wählen, dann können Sie alle internen AMAL-Register ansprechen. In diesen Modus stellt n die Register R0 bis R9 dar und liegt zwischen 0 und 9.

Das folgende Beispiel zeigt Ihnen, wie Sie die gegenwärtige X-Position eines Sprites vom Basic aus ermitteln können:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,100,100,1
A$="Loop: Let RX=X+1; LetX=RX; Pause; Jump Loop"
Rem X läuft über, wenn >640
Amal 1,A$ : Amal On : Curs Off
Do
 Locate 0,0
 Z=Asc("X")-65 : Rem Beachte den Einsatz von ASC zur Ermittlung der Register Nummer
 Print Amreg(Asc("X")-65)
Loop
```

AMPLAY *(Steuert eine durch PLay erzeugte Animation)*

AMPLAY Tempo,Richtung[Start TO Ende)]

Alle Bewegungsfolgen, die Sie mit dem AMAL PL-Befehl hervorgerufen haben, werden durch die internen Register R0 und R1 gesteuert. Dabei wird jedem Objekt sein eigener Satz von AMAL-Registern zugewiesen. Wenn Sie also mehrere Objekte animieren, dann müssen Sie oft eine Reihe dieser Register mit genau denselben Werten laden.

Das können Sie zwar mit der normalen AMREG-Funktion erreichen, aber es wäre natürlich mit einer einzigen Anweisung, durch die Sie R0 und R1 für eine ganze Gruppe von Objekten gleichzeitig verändern können, wesentlich leichter. Und genau das ist der Sinn und Zweck der AMPLAY-Anweisung.

AMPLAY erfaßt *Tempo* und *Richtung* Ihrer Bewegung und lädt sie in die Register R0 und R1 in den gewählten Kanälen.

Tempo steuert die Geschwindigkeit Ihres Objektes auf dem Bildschirm. Dieser Parameter stellt eine Verzögerung in Fünfzigstelsekunden für jeden folgenden Bewegungsschritt ein.

Richtung verändert die Richtung der Bewegung. Hier ist eine Liste mit dem verschiedenen Möglichkeiten:

Wert	Bewegungsrichtung
>0	Bewegt das gewählte Objekt in die ursprüngliche Bewegungsrichtung.
0	Kehrt die Bewegung um und bewegt das Objekt rückwärts.
-1	Bricht den Bewegungsablauf ab und springt zur nächsten Anweisung in Ihrer AMAL-Animationssequenz.

Als Default wirkt diese Anweisung auf alle aktuellen Animationskanäle. Das können Sie ändern, indem Sie ausdrückliche *Start*- und *End*punkte in der Anweisung angeben. *Start* stellt die Kanalnummer des ersten Objektes dar und *Ende* ist die Kanalnummer des letzten Objektes auf Ihrer Liste.

Beachten Sie, daß Sie je nach Bedarf die Parameter *Tempo* und *Richtung* auch weglassen können. Hier sind einige Beispiele:

```
Amplay ,0 : Rem Dreht die Objekte um
Amplay 2, : Rem Verlangsamt die Bewegungsmuster
Amplay 3,1 : Rem Stellt Tempo auf drei und Richtung auf 1
Amplay ,-1 3 To 6 : Rem Stop Bewegungen in Kanälen 3,4,5 und 6
```

=CHANAN *(Überprüft AMAL-Animation)*

s=CHANAN(Kanal)

Dies ist eine einfache Funktion, die den Status der AMAL-Animationssequenz überprüft und -1 (richtig) ausweist, wenn sie derzeit aktiv ist oder 0 (falsch), wenn die Animation völlig abgelaufen ist. *Kanal* enthält die Nummer des Kanals, der überprüft werden soll. Hier ist ein Beispiel dazu:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,150,11
M$="Anim 12,(11,4)(12,4)(13,4)(14,4)(15,4)(15,4);"
Amal 9,M$ :Amal On
While Chanan(9)
Wend
Print "Animation beendet"
```

=CHANMV *(Überprüft, ob sich ein Objekt noch bewegt)*

s=CHANMV(Kanal)

Weist den Wert -1 (richtig) aus, wenn sich das dem channel zugewiesene Objekt derzeit noch bewegt, sonst erscheint 0 (falsch).

Dieser Befehl kann in Verbindung mit der AMAL Move-Anweisung eingesetzt werden, um zu überprüfen, ob für eine Bewegungsfolge noch genug Platz vorhanden ist. Ist das nicht der Fall, so können Sie die Bewegungsfolge dann an einer geeigneteren Position mit der entsprechenden Zeichenkette neu ansetzen. Dazu folgendes Beispiel:

```
Load "AMOS_DATA:Sprites/Monkey_right.abk" : Get Sprite Palette
Sprite 9,150,50,11
M$="Move 300,150,150; Move -300,-150,75"
Amal 9,M$ : Amal On
While Chanmv(9)
Wend
Print "Bewegung abgeschlossen"
```

AMAL-Fehler

=AMALERR *(Weist die Position eines Fehlers aus)*

p=AMALERR

AMALERR weist in der aktuellen Animationszeichenkette die Position aus, an der ein Fehler gemacht wurde. Wenn Sie diese Zeichenkette genau untersuchen, werden Sie Ihre Fehler schnell korrigieren können. Hier ist ein Beispiel dafür:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Sprite 8,100,100,1
A$="L: IF X=300 then Jump L else X=X+1; Jump L"
Amal 8,A$
```

Dieses Programm ruft einen Syntaxfehler hervor, denn "IF" wird als die beiden Anweisungen "I" und "F" interpretiert. Um die Position dieses Fehlers in der Animationskette zu finden, geben Sie im Direktfenster die folgende Anweisung ein:

```
Print Mid$(A$,Amalerr,Amalerr+5)
```

Fehlermeldungen

Wenn Sie in einem Ihrer AMAL-Programme einen Fehler machen, dann geht AMOS mit der entsprechenden Fehlermeldung ins Basic zurück. Hier ist eine komplette Aufstellung der Fehlermeldungen, die durch dieses System hervorgerufen werden können, zusammen mit einer Erklärung der möglichen Ursachen.

Bank nicht reserviert

Dieser Fehler wird hervorgerufen, wenn Sie versuchen, die PLay-Anweisung aufzurufen, ohne zuvor eine Bank in den Speicher zu laden, die die Bewegungsdaten enthält. Diese Bank erstellen Sie mit Hilfe des speziellen AMAL-Zusatzprogramms. Wenn Sie PLay nicht verwendet haben, dann überprüfen Sie, ob Sie die Anweisungen **P**ause und **L**et in Ihrem Programm entsprechend voneinander abgetrennt haben.

Befehl läuft nur im Autotest

Sie haben aus Ihrem AMAL-Hauptprogramm versehentlich entweder die Direct- oder eXit-Anweisung aufgerufen.

Falscher Befehl innerhalb Autotest

Autotest kann nur in Verbindung mit einer begrenzten Reihe von AMAL-Anweisungen eingesetzt werden. Sie können Ihre Objekte in einem Autotest überhaupt nicht bewegen. Überprüfen Sie also, ob falsche Befehle wie Move, Anim oder For...Next aufgerufen wurden.

Sprung in/aus Animationskette nicht erlaubt

Die Befehle in der Autotest-Funktion laufen völlig getrennt von Ihrem AMAL-Hauptprogramm. Es ist also in AMAL nicht zulässig, direkt in eine AUtotest-Prozedur zu springen. Um einen Autotest zu verlassen und zu Ihrem AMAL-Hauptprogramm zurückzukehren, müssen Sie entweder Direct oder eXit verwenden.

Sprungmarke schon in Animationskette definiert

Sie haben versucht in Ihrem AMAL- Programm dieselbe Sprungmarke doppelt zu definieren. Alle AMAL-Sprungmarken bestehen nur aus einem einzigen Großbuchstaben. Also sind die Bezeichnungen **Test** und **Total** nur zwei verschiedene Schreibweisen einer einzigen Sprungmarke (T). Diese Fehlermeldung erscheint auch, wenn Sie zwei Anweisungen durch einen Doppelpunkt ":" getrennt haben. Sie sollten hier stattdessen einen Strichpunkt verwenden.

Sprungmarke in Animationskette nicht definiert

Dieser Fehler entsteht, wenn Sie versuchen, zu einer Sprungmarke zu springen, die gegenwärtig in Ihrer Animationskette nicht existiert.

Next ohne For in Animationskette

Wie im Basic muß auch hier jede For- Anweisung mit dem entsprechenden Next gepaart sein. Überprüfen Sie vor allem die geschachtelten Schleifen nach einem einsamen Next-Befehl.

Syntax Fehler in Animationskette

Sie haben sich in einer Ihrer Animationsketten vertippt. Dieser Fehler entsteht oft, wenn eine AMAL-Anweisung wie der entsprechende Basic-Befehl in voller Länge eingegeben wurde. Vergessen Sie nicht, daß AMAL-Befehle nur aus einem oder zwei Großbuchstaben bestehen. Wenn Sie also versuchen, Ihre Anweisungen als FOR und NEXT einzugeben, erhalten Sie eine Fehlermeldung. Die korrekte Syntax dieser Befehle lautet For...Next.

Animationskanäle

Mit AMOS können Sie bis zu 64 verschiedene AMAL-Programme gleichzeitig ablaufen lassen. Jedem Programm wird dabei ein bestimmter Animationskanal (*channel*) zugewiesen.

Nur bei den ersten 16 Kanälen können Sie mit Interrupts arbeiten. Wenn Sie mehr Objekte animieren möchten, müssen Sie die Interrupts mit der Anweisung SYNCHRO OFF abschalten. Sie können die AMAL-Programme jetzt Schritt für Schritt ausführen, indem Sie in Ihrer Hauptprogrammschleife den SYNCHRO-Befehl ausdrücklich aufrufen. Als Default werden alle Interrupt-Kanäle den entsprechenden Hardware-Sprite zugewiesen.

CHANNEL *(Weist einem AMAL-Kanal ein Objekt zu)*

CHANNEL n TO ObjeKt s

Durch die CHANNEL-Anweisung wird einem Animationskanal ein bestimmtes Bildschirm*objekt* zugewiesen. In AMAL sind Sie jedoch nicht auf einen einzigen Kanal pro Objekt begrenzt. Falls nötig, kann jedes einzelne Objekt auf dem Bildschirm problemlos mit mehreren Kanälen animiert werden. Es gibt eine Reihe verschiedener Ausführungen dieser Anweisung.

Animieren eines berechneten Sprites

CHANNEL n TO SPRITE s

Hier wird das Sprite mit der Nummer *s* dem Kanal *n* zugewiesen. Als Default werden die Kanäle 0 bis 7 den entsprechenden Hardware-Sprites automatisch zugewiesen. Die Kanäle 8 bis 15 sind für die zugehörigen berechneten Sprites reserviert.

Um nun die berechneten Sprites über 16 zu animieren, müssen Sie sie einem Animationskanal mit dem CHANNEL-Befehl direkt zuweisen. Wie üblich bezeichnen Sprite-Nummern von 8 bis 63 berechnete Sprites anstatt einfacher Hardware- Sprites. Hier ist ein Beispiel:

```
Channel 5 To Sprite 8 : Rem Animiert berechnetes Sprite 8 über Kanal 5.
```

Die X,Y-Register in Ihrem AMAL-Programm beziehen sich jetzt auf die Hardware-Koordinaten des jeweiligen Sprites. Entsprechend enthält das Register A das aktuelle Sprite-Bild.

Das Animieren eines Blitter-Objektes

AMAL-Programme können auch dazu eingesetzt werden, um Blitter-Objekte zu animieren.

CHANNEL n TO BOB b

Hier wird das Blitter-Objekt b dem Animationskanal n zugewiesen. Dieses Objekt wird jetzt genauso behandelt wie das entsprechende Hardware-Sprite. Der einzige Unterschied besteht darin, daß die Register X und Y jetzt die Position Ihres BOBs in **Bildschirm** koordinaten enthalten.

Es ist jedoch Vorsicht geboten, wenn Sie das Austauschen der Schirme mit dem Befehl DOUBLE BUFFER aktiviert haben, denn dann wird dieses System automatisch für alle BOB-Animationen verwendet. Ein umfassendes Beispiel zu diesem Befehl finden Sie in **8** im MANUAL-Unterverzeichnis.

Das Bewegen eines Schirms

AMOS-Basic ermöglicht es Ihnen, Ihren Schirm beliebig auf dem Fernsehschirm zu positionieren. Normalerweise wird das durch die Anweisung SCREEN DISPLAY gesteuert. Manchmal ist es jedoch nützlich, den Schirm durch Interrupts zu bewegen.

CHANNEL n TO SCREEN DISPLAY d

Mit dieser Anweisung wird der Kanal mit der Nummer *n* auf den Schirm *d* eingestellt. Der Schirm *d* kann an jeder beliebigen Stelle in Ihrem Programm definiert worden sein. Sie erhalten jedoch eine Fehlermeldung, wenn dieser Schirm nicht eröffnet wurde, bevor Sie Ihre Animation starten.

Die X- und Y-Variablen in AMAL enthalten jetzt die Position Ihres Schirms in Hardware-Koordinaten. Das Register **A** wird in dieser Option nicht eingesetzt und Sie können Ihre Schirm nicht mit dem Befehl Anim animieren. Davon abgesehen können jedoch alle normalen AMAL-Anweisungen wie üblich ausgeführt werden. Sie können mit diesem System Ihr Bild also gut über den Bildschirm springen lassen. Hier sind zwei Beispiele dazu:

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",1
Channel 0 To screen display 1
Amal 0,"Loop:Move 0,200,100 ; Move 0,-200,100 ; Jump Loop"
Amal On 0 : Direct
```

```
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF",1
Limit Mouse 100,50 to 430,330
Channel 0 To screen display 1
Rem Schirm kann nur an bestimmten X Positionen angezeigt werden
Amal 0,"Loop: Let X=XM; Let Y=YM; Jump Loop"
Amal On : Direct
```

Um ein weiteres Beispiel für diese Technik zu sehen, können Sie **BEISPIEL 14.4** aus dem MANUAL-Unterverzeichnis laden. Hier sehen Sie, wie SCREEN DISPLAY in Verbindung mit den Menü-Befehlen eingesetzt werden kann, um den Menü-Schirm auf Ihrem Display nach oben oder unten zu schieben. Diese Technik weist große Ähnlichkeiten zu dem Display-System auf, das in den ausgezeichneten Abenteuerspielen von Magnetic Scrolls eingesetzt wird.

Das Hardware-Scrollen

Das Hardware-Scrollen kann zwar mit dem AMOS Basic-Befehl SCREEN OFFSET durchgeführt werden, es ist aber oft wesentlich leichter, wenn Sie Ihre Schirme stattdessen mit AMAL animieren, denn so entsteht ein viel glatterer Effekt.

CHANNEL n TO SCREEN OFFSET d

Auf diese Art wird das AMAL-Programm mit der Nummer *n dem Schirm* d für das

Hardware-Scrollen zugewiesen. Die Register X und Y beziehen sich jetzt auf den Abschnitt des Schirms, der auf Ihrem Fernsehschirm dargestellt werden soll. Wenn Sie nun diese Register vertauschen, wird der sichtbare Schirmbereich auf dem Display gescrollt. Dazu folgendes Beispiel:

```
Screen Open 0,320,500,32,Lowres : Rem Öffnet Schirm mit Überhöhe
Screen Display 0,,45,320,250
Load Iff "AMOS_DATA:IFF/AMOSPIC.IFF"
Screen Copy 0,0,0,320,250, To 0,0,251
Screen 0 : Flash Off: Get Palette (0)
Channel 0 to Screen Offset 0
Amal 0,"Loops: Let X=XM-128; Let Y=YM-45; Pause; Jump Loop"
Amal On : Wait Key
```

Dieses Programm ermöglicht es Ihnen, mit der Maus durch den Schirm zu scrollen. Versuchen Sie die Maus im Direktmodus zu bewegen. Ein weiteres Beispiel für das Hardware-Scrollen finden Sie in **BEISPIEL 14.5**.

Das Verändern der Schirmgröße

CHANNEL n TO SCREEN SIZE s

Mit dieser Anweisung können Sie die Größe eines Schirms durch AMAL verändern. *s* ist die Nummer des Schirms, der manipuliert werden soll. Die Register X und Y steuern jetzt jeweils Breite und Höhe Ihres Schirms. Sie entsprechen in diesem Fall den Parametern W und H in der Anweisung SCREEN DISPLAY. Hier ist ein Beispiel:

```
Load Iff "AMOS_DATA:IFF/Magic_Screen.IFF",0
Channel 0 To Screen Size 0
Screen Display 0,,,320,1 : Rem Stellt Schirmgröße auf 1
A$="Loop: For R0=0 To 255 ; Let Y=R0 ; Next R0;"
A$=A$+"For R0=0 To 254; Let Y=255-R0; J Loop"
Amal 0,A$ : Amal On : Direct
```

Regenbogen

CHANNEL n TO RAINBOW r

Mit dieser Option können Sie in einem AMAL-Programm einen Regenbogen erzeugen. Wie üblich stellt *n* die Nummer des Animationskanal dar und liegt zwischen 0 und 63. *r* ist die Identifikationsnummer Ihres Regenbogens (0-3).

X enthält die aktuelle BASIS Ihres Regenbogens. Das ist die erste Farbe Ihrer Regenbogenpalette, die angezeigt wird. Wenn Sie sie verändern, scheint es, als ob sich der Regenbogen dreht. Y gibt die Linie auf dem Bildschirm an, an der der

Regenbogeneffekt beginnt. Wenn Sie diesen Wert verändern, bewegt sich der Regenbogen nach oben oder unten. Alle Koordinaten werden im Hardware-Format angegeben.

Das Register A speichert die Höhe Ihres Regenbogens auf dem Bildschirm. Sie finden eine Demonstration dieses Systems in 11. Weiter Einzelheiten hierzu können Sie unter dem RAINBOW-Befehl in AMOS-Basic nachlesen.

Techniken für Fortgeschrittene

Das AUTOTEST-System

Normalerweise werden alle AMAL-Programme streng der Reihe nach von Anfang bis Ende durchgeführt. Es ist dabei unvermeidlich, daß das Ausführen mancher Befehle wie zum Beispiel Move oder For...Next ein paar Sekunden dauert. Meistens ist das kein Problem, aber beim Ablaufen einiger Programme kann es doch zu beträchtlichen Verzögerungen führen. Nehmen wir zum Beispiel einmal folgendes einfaches Programm:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 8,130,50,1
Amal 8,"Loop: Let R0=XM-X; Let R1=YM-Y; Move R0,R1,50; Jump Loop"
Amal On : Direct
```

Wenn Sie die Maus bewegen, soll ihr nun das Sprite auf dem Bildschirm folgen.In der Praxis ist die Antwortzeit aber etwas lange, denn die neuen Werte für XM und YM werden erst eingegeben, wenn die Bewegung des Sprites völlig abgeschlossen ist. Versuchen Sie einmal, die Maus im Kreis zu bewegen. Jetzt weiß der arme Tintenfisch garnicht mehr, was los ist!

Dieses Problem wird durch Autotest gelöst, denn jetzt werden Ihre Abfragen am Anfang jedes VBLs durchgeführt, bevor das aktuelle Programm weitergeht. Ihre Abfragen erfolgen jetzt in regelmäßigen Abständen von 1/50 Sekunden und führen daher praktisch unverzüglich zu einer Reaktion.

Die Autotest-Befehle

Die Autotest-Syntax lautet folgendermaßen:

AUtotest *(Tests)*

Tests kann einen der folgenden AMAL-Befehle darstellen:

Let

L Reg=Exp

Das ist die normale AMAL Let-Anweisung. Sie weist dem Register *Reg* das Ergebnis eines Ausdrucks (*Exp*) zu.

Jump

Jump Label

Mit dem Jump-Befehl können Sie zu einem anderen Teil des aktuellen Autotests springen. Die Sprungmarke *Label* wird durch einen Doppelpunkt definiert und **muß** in den Autotest-Klammern eingeschlossen sein.

eXit

Verläßt den Autotest und kehrt an den Punkt in das Hauptprogramm zurück, an dem unterbrochen wurde.

Wait

Wait schaltet das AMAL-Hauptprogramm völlig ab und führt nur den Autotest durch.

If

Um den Testprozeß in einer Autotest-Routine zu vereinfachen, gibt es eine besondere, erweiterte Version der AMAL If-Anweisung. So können Sie, je nachdem, wie das Ergebnis des Ausdrucks *Exp* lautet, einen der drei möglichen Wege einschlagen.

if Exp Jump L	(Springt zu einem anderen Teil des Autotests)
if Exp Direct L	(Wählt den Teil des Programmes, der nach dem Autotest ausgeführt werden soll)
if Exp eXit	(Verläßt Autotest)

On

ON nimmt das Hauptprogramm nach einer vorhergegangenen Wait-Anweisung wieder auf. So können Sie zum Beispiel auf ein bestimmtes Ereignis wie zum Beispiel einen Maus-Klick warten, ohne wertvolle Rechnerzeit zu verschwenden.

Direct

Direct Label

Durch Direct können Sie den Punkt, an dem Ihr Hauptprogramm nach Ihrem Test wieder aufgenommen wird, verändern. Jetzt springt AMAL beim nächsten Vertical Blank

automatisch zu diesem Punkt. Beachten Sie dabei, daß *label* hier außerhalb der Autotest-Klammern definiert werden muß.

Im Autotest

Hier ist nun eine neue Fassung des vorher dargestellten Beispiels, erstellt unter Verwendung der Autotest-Funktion:

```
Load "AMOS_DATA:Sprites/octopus.abk"
Sprite 8,130,50,1 : Get Sprite Palette
A$="AUtotest (If R0<>XM Jump Update"
A$=A$+"If R1<>YM Jump Update else eXit"
A$=A$+"Update: Let R0=XM; Let R1=YM; Direct M)" : Rem Ende des Autotests
A$=A$+"M: Move R0-X,R1-Y,20 Wait;" : Rem Versuche für 20 verschiedene Werte einzusetzen
Amal 8,A$ : Amal On
```

Das Sprite folgt Ihrer Maus jetzt auf dem Fuße, ganz egal, wie schnell Sie die Maus bewegen. Dieses Programm funktioniert folgendermaßen:

Alle 50stel Sekunden werden die Mauskoordinaten mit den Funktion XM und YM überprüft. Wenn sie sich seit der letzten Abfrage nicht geändert haben, wird Autotest mit den eXit-Befehl abgebrochen. Das Hauptprogramm wird jetzt genau dort wieder aufgenommen, wo es unterbrochen wurde.

Wenn die Maus jedoch bewegt wurde, dann beginnt die Autotest-Routine wieder am Anfang mit dem Hauptprogramm (Sprungmarke **M**) unter Verwendung der neuen Koordinaten in XM und YM.

Überlegungen zum Timing

UPDATE EVERY

(Spart etwas Zeit für Ihre Basic-Programme)

UPDATE EVERY n

Zwar werden die meisten AMAL-Programme praktisch sofort ausgeführt, aber die Objekte, die sie manipulieren, müssen trotzdem auf dem Amiga-Bildschirm extra neu gezeichnet werden.

Man kann schlecht vorhersagen, wieviel Zeit diese Aktualisierungsprozedur benötigt, und es kann sich im Laufe Ihres Programmes auch ändern. Das kann zu einem störenden Zittern im Bewegungsablauf bestimmter Objekte führen.

Der Befehl UPDATE EVERY verlangsamt den Aktualisierungsprozeß nun, damit selbst das größte Objekt während einer einzigen Bildschirmaktualisierung neu gezeichnet werden kann. Auf diese Art wird das Animationssystem angepaßt und es entstehen wunderbar glatte, fließende Bewegungen.

n ist die Anzahl der Vertikal Blanks (50stel Sekunden), die zwischen jedem Aktualisieren des Bildschirms liegen. In der Praxis sollten Sie mit dem Wert zwei beginnen, und ihn dann Schritt für Schritt erhöhen, bis die Bewegung ganz glatt ist.

Eine nützliche Nebenwirkung von UPDATE EVERY liegt darin, daß Sie mehr Zeit

für die Ausführung Ihrer Programme durch das Basic reservieren können. Durch umsichtiges Einsetzen dieser Anweisung können Sie Ihre Programme bis zu 30% schneller machen, ohne Ihre glatten Animationssequenzen zu zerstören.

Das Umgehen der Begrenzung auf 16 Objekte

SYNCHRO *(Führt ein AMAL-Programm direkt aus)*

SYNCHRO [ON/OFF]

Normalerweise können Sie mit AMOS-Basic bis zu 16 verschiedene AMAL- Programme gleichzeitig ablaufen lassen. Diese Begrenzung ergibt sich aus der Gesamtgeschwindigkeit der Hardware des Amiga. Jedes AMAL-Programm nimmt einen Teil der verfügbaren Rechnerzeit für sich in Anspruch. Wenn Sie also das normale Interrupt-System einsetzen, bleibt nur gerade genug Zeit um 16 verschiedene Programme auszuführen.

Durch den SYNCHRO-Befehl können Sie diese Beschränkung überschreiten, indem Sie Ihre AMAL-Programme direkt aus dem Basic ausführen. Anstelle der Interrupts werden alle AMAL-Programme jetzt auf einmal durch den SYNCHRO-Befehl aufgerufen. Da AMAL-Programme schneller ablaufen als die entsprechenden Basic-Routinen, sind Ihre Animationen jetzt wunderbar glatt. Und Sie können nun entscheiden, wann und wo Ihre AMAL-Routinen in Ihrem Programm ausgeführt werden.

Ein zusätzlicher Vorteil besteht darin, daß Sie jetzt Befehle zur Kollisionsabfrage wie zum Beispiel Bob Col oder Sprite Col direkt in Ihre AMAL-Routinen einbauen können. Im Interrupt-System können diese Befehl nicht verwendet werden, da sie den Blitter-Chip des Amiga einsetzen. In Verbindung mit Interrupts wäre das völlig unmöglich.

Bevor Sie jedoch SYNCHRO aufrufen, müssen Sie erst die Interrupts mit SYNCHRO OFF abschalten. Das sollten Sie unbedingt tun, bevor Sie Ihre AMAL-Programme definieren, sonst erhalten Sie eine Fehlermeldung, wenn Sie einen der Kanäle über 15 verwenden möchten.

Aufgrund der reinen Leistung des Animationssystems können Sie ganze Ballerspiele fast ausschließlich in AMAL schreiben. Für Ihr Basic-Programm bleiben dann nur die einfachen Aufgaben, wie das Verwalten der Punktstandtabellen und das Laden der Angriffswellen von der Diskette. Das Ergebnis ist vom reinen Maschinencode so gut wie nicht zu unterscheiden. Ein gutes Beispiel dafür ist das Spiel Cartoon Capers, das erste kommerzielle Spiel auf dem Markt, das vollständig in AMOS geschrieben wurde.

Sie finden in **BEISPIEL 14.6** im MANUAL-Unterverzeichnis eine Demonstration des SYNCHRO-Befehls.

STOS-kompatible Animationsbefehle

Die Originalversion von STOS-Basic enthielt ein leistungsstarkes Animationssystem, mit dem Sie Ihre Sprites mittels Interrupts in ziemlich komplexen Abläufen bewegen konnten. Damals wurden diese Befehle als bahnbrechend bejubelt.

Obwohl sie jetzt vom AMAL-System in den Schatten gestellt wurden, stellen sie doch eine einfache Einführung in die Animation auf dem Amiga dar. Deshalb bietet Ihnen AMOS das gesamte STOS-Animationssystem als Zugabe!

Wenn Sie STOS-Programme in AMOS umsetzen möchten, sollten Sie die folgenden Punkte beachten:

- Im Unterschied zu STOS können die Bewegungsabläufe in AMOS-Basic jedem beliebigen Animationskanal zugewiesen werden. Daher können die Move- Befehle verwendet werden, um BOBs, Sprites oder Schirme mit genau derselben Technik zu bewegen.

  Als Default werden alle Animationskanäle den entsprechenden Hardware-Sprites zugeordnet. In der Praxis ersetzten Sie sie aber vielleicht lieber durch Blitter- Objekte, da diese den normalen STOS Basic-Sprites ähnlicher sind. Fügen Sie am Anfang Ihres Programms eine Reihe von CHANNEL-Befehlen folgendermaßen ein:

```
Channel 1 to bob 1
Channel 2 to bob 2
:    :    :
```

  Vergessen Sie nicht, während Ihrer Initialisierungs-Prozedur DOUBLE BUFFER aufzurufen, sonst flimmern Ihre Bobs ganz entsetzlich, wenn sie bewegt werden.

- Sie können sowohl für STOS- wie auch AMAL-Programme dieselben Kanäle verwenden. Es ist also einfach, die Programme zu erweitern, wenn Sie sie einmal zu AMOS-Basic konvertiert haben. Die Reihenfolge der Ausführung lautet:

```
AMAL
MOVE X
MOVE Y
ANIM
```

MOVE X *(Bewegt ein Sprite horizontal)*

MOVE X n,m$

MOVE X definiert eine Reihe von horizontalen Bewegungen, die in der Folge auf dem Animationskanal mit der Nummer *n* ausgeführt werden.

n liegt im Bereich zwischen 0 und 15 und bezieht sich auf ein Objekt, das Sie zuvor mit dem CHANNEL-Befehl zugeordnet haben. m$ enthält eine Reihe von Anweisungen, die zusammen genommen sowohl Geschwindigkeit wie auch Richtung Ihres Objektes bestimmen. Diese Befehle werden von Klammern eingefaßt und in folgendem Format eingegeben:

(Geschwindigkeit,Schritte,Anzahl)

Die Anzahl der Befehle, die eine Bewegungskette enthalten kann, ist nur durch den verfügbaren Speicherplatz begrenzt.

Geschwindigkeit stellt eine Verzögerung in 50stel Sekunden zwischen den einzelnen Bewegungsschritten ein. Die Geschwindigkeit kann von 1 (sehr schnell) bis 32767 (unwahrscheinlich langsam) variieren.

Schritte gibt die Anzahl der Pixel an, die das Objekt bei jedem Schritt bewegt wird. Wenn Schritte positiv ist, bewegt sich das Sprite nach rechts, und bei einem negativen Wert geht es nach links.

Die offensichtliche Geschwindigkeit des Objektes hängt von der Kombination aus Geschwindigkeit (speed) und Schrittgröße ab. Große Schritte verbunden mit einer gemäßigten Geschwindigkeit bewegen das Objekt rasch, aber stockend über den Bildschirm. Entsprechend wird bei einer kleinen Schrittgröße kombiniert mit einer hohen Geschwindigkeit das Objekt ebenfalls schnell bewegt, die Bewegung ist hier jedoch wesentlich glatter. Die höchsten Geschwindigkeiten können erzielt werden, wenn die Schrittgröße ungefähr 10 (bzw. -10) beträgt.

Anzahl bestimmt, wie oft die Bewegung wiederholt wird. Die zulässigen Werte liegen zwischen 0 und 32767. Bei dem Wert 0 wird der Bewegungsablauf unablässig wiederholt.

Zusätzlich zu den obengenannten Befehlen können Sie am Ende Ihrer Bewegungskette auch eine der folgenden Anweisungen anhängen. Die wichtigste dieser Erweiterungen ist die L-Anweisung (für Schleife), die an den Anfang der Kette zurückspringt und die gesamte Sequenz noch einmal von vorn ablaufen läßt. Hier ist ein Beispiel dafür:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Sprite 1,130,100,1 : Rem Definiert Sprite 5
Move X 1,"(1,5,60)(1,-5,60)L"
Move On
Direct
```

Mit der E-Option können Sie Ihr Objekt anhalten, wenn es einen bestimmten Punkt auf dem Bildschirm erreicht hat. Verändern Sie die zweitletzte Zeile im obenstehenden Beispiel zu:

```
Move X 1,"(1,5,30)E100"
```

Beachten Sie dabei, daß diese Endpunkte nur funktionieren, wenn die X-Koordinate des Objektes genau den Wert erreicht, den Sie in der Anweisung angegeben haben. Wenn dieser Wert ungünstig gewählt ist, dann springt Ihr Objekt möglicherweise über den Endpunkt hinweg und der Schritt schlägt fehl. Dazu folgendes Beispiel:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Channel 2 To Sprite 10
```

```
Print At(0,5)+"Schleife OK"
Sprite 8,130,100,1
Move X 1,"(1,10,30)(1,-10,30)L"
Move On
Print At(o,10)+"Jetzt drücke eine Taste" : Wait Key
Sprite 10,140,150,2
Move X 2,"(1,15,20)L" : Move On 2
Print At(0,15)+"Oh Schreck!" : Wait Key
```

MOVE Y *(Bewegt ein Objekt vertikal)*

MOVE Y n,m$

Diese Anweisung ergänzt den Befehl MOVE X, denn nun können Sie ein Objekt vertikal auf dem Bildschirm bewegen. Wie zuvor bezeichnet *n* auch hier die Nummer einer Animationssequenz, die Sie mit dem CHANNEL-Befehl zugewiesen haben. *n* liegt zwischen 0 und 15.

m$ enthält eine Bewegungskette im gleichen Format wie bei MOVE X. Positive Werte entsprechen einer Bewegung nach unten und negative Werte ergeben eine Bewegung nach oben. Hier sind Beispiele dazu:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,130,10,1 : Rem Install sprite
Move Y 1,"10(1,1,180)L" : Rem Endlosschleife für Sprite von 10,10 bis 190,10
Channel 2 To Screen Display 0 : Rem Weist Schirmposition Kanal 2 zu
Move Y 2,"(1,4,25)(1,-4,25)" : Rem Bewegt Schirm auf und ab
Move On : Wait Key
```

MOVE ON/OFF *(Beginnt/stoppt die Bewegung)*

MOVE ON/OFF [n]

Bevor Ihre Bewegungsabläufe ausgeführt werden, müssen Sie sie mit dem Befehl MOVE ON aktivieren.

n bezieht sich auf die Animationssequenz, die Sie starten möchten, und liegt im Bereich von 0 bis 15. Wenn Sie diesen Parameter weglassen, dann werden alle Bewegungen gleichzeitig aktiviert.

MOVE OFF hat genau die entgegengesetzte Wirkung - diese Anweisung bringt die entsprechenden Bewegungsfolgen zum Stillstand.

MOVE FREEZE *(Zeitweise Unterbrechung der Sprite-Bewegungen)*

MOVE FREEZE [n]

Die Anweisung MOVE FREEZE hält die Bewegung von einem oder mehreren Objekten auf dem Bildschirm zeitweise an. Diese Objekte können danach mit MOVE ON wieder

aktiviert werden.

n kann wahlweise eingegeben werden und gibt die Nummer eines bestimmten Objektes an, das durch diese Anweisung zeitweise angehalten werden soll.

=MOVON *(Weist den Bewegungsstatus aus)*

x=MOVON(n)

MOVON überprüft, ob ein bestimmtes Objekt durch die Anweisungen MOVE X oder MOVE Y gerade bewegt wird. -1 (richtig) wird ausgewiesen, wenn das Objekt tatsächlich in Bewegung ist, und 0 (falsch), wenn es stillsteht. Verwechseln Sie diesen Befehl bitte nicht mit der Anweisung MOVE ON.

Beachten Sie, daß MOVON nur nach Bewegungsabläufen sucht, die durch einen der MOVE-Befehle in Gang gesetzt wurden. Durch AMAL generierte Animationen können auf diese Art nicht erfaßt werden.

ANIM *(Animiert ein Objekt)*

ANIM n,a$

Anim schiebt ein Objekt automatisch durch eine Reihe von Bildern und erzeugt so einen glatten Animationseffekt auf dem Bildschirm. Diese Animationen werden 50mal pro Sekunden durch Interrupts ausgeführt, und können also gleichzeitig mit Ihren Basic-Programmen ablaufen. n ist die Nummer des Kanals, so wird angegeben, welches Sprite oder BOB durch diese Anweisung animiert werden soll.

a$ enthält eine Reihe von Anweisungen, die Ihre Animationssequenz definieren. Jede Operation wird in zwei getrennte Komponenten aufgeteilt, die in runde Klammern gestellt werden.

Bild ist die Nummer des Bildes, das in jedem Rahmen der Animation dargestellt werden soll. *Verzögerung* gibt die Zeitspanne an, für die dieses Bild auf dem Bildschirm erscheint (in 50stel Sekunden). Dazu folgendes Beispiel:

```
Load "AMOS_DATA:Sprites/octopus.abk" : Get Sprite Palette
Channel 1 To Sprite 8 : Sprite 8,200,100,1
Anim 1,"(1,10)(2,10)(3,10)(4,10)"
Anim On : Wait Key
```

Genau wie beim MOVE-Befehl gibt es auch hier eine L-Anweisung, durch die Sie Ihre Animationen ständig wiederholen können. Verändern Sie im obigen Beispiel dazu nur den ANIM-Befehl folgendermaßen:

```
Anim 1,"(1,10)(2,10)(3,10)(4,10)L"
```

ANIM ON/OFF *(Beginnt eine Animation)*

ANIM ON/OFF [n]

ANIM ON aktiviert eine Reihe von Animationen, die zuvor mit dem ANIM-Befehl erzeugt wurden. *n* gibt die Nummer einer bestimmten Animationssequenz an, die initialisiert werden soll. Wenn Sie diesen Parameter weglassen, werden alle Animationssequenzen sofort begonnen.

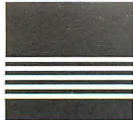
ANIM OFF [n]

Diese Anweisung hält eine oder mehrere Animationssequenzen an, die durch ANIM ON aktiviert wurden.

ANIM FREEZE *(Unterbricht eine Animation)*

ANIM FREEZE [n]

ANIM FREEZE unterbricht die aktuelle Animationssequenz auf dem Bildschirm zeitweise. *n* kann eine bestimmte Animationssequenz angeben, die angehalten werden soll. Wenn Sie *n* weglassen, werden alle gegenwärtigen Animationen unterbrochen. Sie können sie jederzeit wieder aufnehmen, indem Sie einfach ANIM ON aufrufen.



15: Hintergrundgrafik

Es ist heutzutage durchaus üblich, daß ein Spielhallen-Spiel Hunderte von verschiedenen Bildschirmen umfaßt. Wenn man es packt, kann man einen Bildschirm mit 32 Farben in einen Speicherplatz von etwa 30 KByte pressen. Also würden 100 Bildschirme etwa 3 Megabyte an Daten entsprechen. Stellen Sie sich vor, wie schwierig es wäre, das alles in einem normalen A500 unterzubringen!

Der klassische Weg, diese Begrenzung zu umgehen, besteht darin, Ihren Hintergrund aus einer Reihe einfacher Bausteine zusammenzusetzen. Wenn Sie diese Bausteine einmal erstellt haben, können Sie sie in jeder beliebigen Reihenfolge auf dem Bildschirm anordnen. Sie können damit also das gleiche Sortiment an Bausteine dazu verwenden, eine Vielzahl verschiedener Schirme zu erzeugen. Jedes Bild wird jetzt einfach als Auflistung seiner Bestandteile gespeichert und verbraucht so nur einen Bruchteil des ursprünglichen Speichers.

Um dieses System voll auszuschöpfen, müssen Sie Ihre verschiedenen Bildschirmmaps natürlich irgendwie definieren. Und was haben wir da zu Ihrer Unterstützung auf der AMOS-Programmdiskette untergebracht? Sie haben es erraten: ein leistungsstarkes Zusatzprogramm für das Definieren der Maps (map definer accessory). In der zugehörigen Dokumentationsdatei finden Sie alle Einzelheiten darüber.

AMOS-Basic enthält außerdem eine Reihe von Anweisungen, die es Ihnen erleichtern, Ihre Bausteine auf den Bildschirm zu zeichnen. So können Sie den Scroll-Hintergrund, der das Erkennungszeichen jedes modernen Spielhallen-Spiels ist, ganz leicht erzeugen.

Icons

Ein Icon ist ein eigenes Bild, das speziell zum Erzeugen Ihrer Hintergründe erstellt wird. Wenn Sie ein Icon einmal gezeichnet haben, ist es immer fest an dieser Stelle angebracht. Sie können es also mit dem AMAL-Animationssystem nicht an eine neue Position verschieben.

Alle Icons werden in ihrer eigenen AMOS-Speicherbank (Bank Nummer 2) gespeichert. Diese Bank wird mit dem Sprite-Definer-Zusatzprogramm (auf der AMOS-Programmdiskette) erzeugt und wird dann automatisch zusammen mit Ihren Basic-Programmen gespeichert.

Wie BOBs werden auch Icons unter Einsatz des erstaunlichen Blitter-Chips angezeigt. Da Icons aber im Grunde statische Objekte sind, werden sie normalerweise im REPLACE-Modus gezeichnet. Deshalb überschreiben Icons alle an der aktuellen Bildschirmposition bereits vorhandenen Grafiken vollständig.

PASTE ICON *(Zeichnet ein Icon)*

PASTE ICON x,y,n

Zeichnet ein Icon mit der Nummer *n* an den GRAFIKKOORDINATEN *x,y* auf den Bildschirm. n ist dabei die Nummer des Icons, das angezeigt werden soll. Es muß allerdings zuvor in der ICON-Bank (Nummer 2) gespeichert worden sein.

Icons können an jeder beliebigen Stelle auf dem Bildschirm positioniert werden. Die üblichen Clipping-Regeln müssen jedoch dabei befolgt werden. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Icons/Map_icons.abk"
Screen Open 0,320,256,32,Lowres : Cls 0 : Get Icon Palette
Rem Zeichnet Rahmen um den Schirm
For X=1 To 11 : Paste Icon X*32,0,1 : Next X
For Y=1 To 6 : Paste Icon 0,Y*32+11,1 : Paste Icon 288,Y*32,1 : Next Y
For X=1 To 11 : Paste Icon X*32,223,1 : Next X
```

Wenn Sie Double-Buffering einsetzen, sollten Sie beachten, daß eine Kopie Ihres Icons sowohl auf den logischen als auch auf den physischen Bildschirm gezeichnet wird. Dies geht natürlich ziemlich langsam, und daher ist es üblich, vor dem Zeichnen der Icons auf den Schirm AUTOBACK 0 aufzurufen. Dadurch werden die Icons auf den aktuellen logischen Schirm beschränkt. Sie können jetzt mit SCREEN COPY den gesamten Hintergrund direkt auf den physischen Bildschirm kopieren und dadurch eine ganze Menge Zeit sparen.

Ein weiteres Beispiel finden Sie im MAPVIEW-Programm auf der AMOS Datendiskette. Hier wird ein Hintergrundbild dargestellt, das Sie mit dem AMOS Map Editor erzeugt haben.

GET ICON *(Erzeugt ein Icon)*

GET ICON [s,] i,tx,ty TO bx,by

Erfaßt ein Bild auf dem Bildschirm und lädt es in das Icon *i*. Wenn dieses Icon derzeit nicht existiert, wird es für Sie in der Bank mit der Nummer 2 erstellt. Falls nötig, wird diese Bank von dem System automatisch reserviert.

i stellt die Nummer Ihres Icons dar und beginnt bei 1. *tx,ty* definieren die obere und untere Ecke eines rechteckigen Bereichs, der die gewünschte Fläche umschließt.

s bestimmt die Nummer des Schirms, der als Quelle Ihres Bildes dient. Wenn Sie diesen Parameter weglassen, wird das Bild stattdessen dem aktuellen Schirm entnommen. Hier ist ein Beispiel für den Einsatz dieses Befehls:

```
Erase 2
F$=Fsel$("*.*","","Lädt einen Schirm"): If F$"" Then Direct
If Exist(f$) Then Load Iff f$,0 Else Direct
```

```
SH=Screen Height : H=SH/32-1 : Sw=Screen Width : W=SW/32-1
For Y=0 To H
  For X=0 To W
      Rem Greift ein Icon
      Get Icon X+Y*W+1,X*32,Y*32 To X*32+31,Y*32+31
  Next X
Next Y
Cls 0
Do
   Paste Icon Rnd(Sw-1),Rnd(SH-1),Rnd(H*W)+1
Loop
```

GET ICON PALETTE *(Erfaßt Icon-Farben)*

GET ICON PALETTE

Erfaßt die Farben der Icon-Bilder in der Bank mit der Nummer 2 und lädt sie in die aktuelle Bildschirmpalette. Dieser Befehl wird normalerweise verwendet, um den Schirm zu initialisieren, nachdem Sie einige Icons von der Diskette geladen haben. Dazu folgendes Beispiel:

```
Rem Lädt einige Icons von der AMOS Datendiskette
Load "AMOS_DATA:Icons/Map_icons.abk"
Get Icon Palette
Paste Icon 100,100,1
```

DEL ICON *(Löscht Icons)*

DEL ICON n [TO m]

Löscht ein oder mehrere Icons aus der Icon-Bank. *n* ist die Nummer des ersten Icons, das entfernt wird.

m kann die Nummer des letzten Icons angeben, das gelöscht werden soll. Wenn Sie diesen Parameter eingeben, dann werden alle Icons, die zwischen den beiden Punkten liegen, nacheinander gelöscht. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Icons/Map_icons.abk"
Paste Icon 100,100,1
Del Icon 1
Paste Icon 100,100,1
```

Nachdem das letzte Icon in einer Bank gelöscht wurde, wird die ganze Bank aus dem Speicher entfernt.

MAKE ICON MASK *(Stellt die Farbe Null als transparent ein)*

MAKE ICON MASK [n]

Normalerweise überschreiben alle Icons, die Sie auf den Bildschirm zeichnen, den vorhandenen Hintergrund vollständig. Das Icon wird dann in einem rechteckigen Feld angezeigt, das durch die Farbe mit der Nummer Null ausgefüllt ist.

Wenn Sie diesen Effekt vermeiden möchten und die aktuellen Grafiken durch Ihre Icons überlagern wollen, dann müssen Sie für die Icons eine Maske erzeugen. Auf diese Art können Sie AMOS mitteilen, daß die Farbe Null transparent sein soll.

n ist dabei die Nummer des Icons, das von dieser Anweisung betroffen ist. Wenn Sie diesen Wert weglassen, dann wird für alle Icons in der Bank eine Maske definiert.

In **BEISPIEL 15.1** im MANUAL-Unterverzeichnis finden Sie eine Demonstration dieses Effekts.

NO ICON MASK *(Entfernt die Maske von einem Icon)*

NO ICON MASK(n)

Entfernt eine zuvor mit MAKE ICON MASK definierte Maske für ein Icon *n*. Siehe NO MASK.

Bildschirmblöcke

Wie Sie ja schon bereits wissen, weist AMOS-Basic eine Reihe von leistungsstarken BLOCK-Befehlen auf, mit denen Sie einen Teil eines Bildes in den Speicher stellen und ihn dann auch beliebig auf dem Bildschirm wieder einsetzen können.

Diese Anweisungen dienen vor allem zur Speicherung temporärer Daten, da Sie die Blöcke nicht zusammen mit Ihren Basic-Programmen speichern können. Blöcke eignen sich besonders gut für die Erstellung von Dialogboxen, da Sie so die Hintergrundbereiche speichern können, bevor Sie Ihre neuen Grafiken anzeigen.

Sie können sie auch gut in Puzzle-Spielen wie Split Personalities einsetzen. Hier können Sie jedes Teil Ihres Bildes in einen Block laden und die Teile dann mit dem Befehl PUT BLOCK auf dem Bildschirm mischen und neu anordnen.

GET BLOCK *(Stellt einen Bildschirmblock in den Speicher)*

GET BLOCK n,ty,ty,w,h,[,Maske]

GET BLOCK bringt einen rechteckigen Bereich in den Nummer *n*, der bei den Koordinaten *tx,ty* beginnt.

n ist die Nummer des Blockes und liegt zwischen 1 und 65535. *tx,ty* sind die Koordinaten der linken oberen Ecke Ihres Blockes. *w,h* enthalten jeweils die Breite und die Höhe des Blockes.

Maske ist ein Flag, durch das Sie wählen, ob für Ihren neuen Block eine Maske erstellt werden soll.

Maske=0	Replace-Modus. Wenn der Block auf den Schirm gezeichnet wird, dann wird er alle an dieser Bildschirmposition bereits vorhandenenGrafiken völlig überschreiben.
Maske=1	Berechnet eine Maske für den Block. Die Farbe Null wird jetzt auf transparent gestellt.

PUT BLOCK *(Kopiert einen zuvor erzeugten Block auf den Bildschirm)*

PUT BLOCK n[,x,y]
PUT BLOCK n,x,y,Planes[,Minterms]

PUT BLOCK kopiert den Block mit der Nummer *n* auf den aktuellen Schirm. *x,y* geben die Position Ihres neuen Blockes auf dem Bildschirm an. Wenn sie weggelassen werden, dann wird der Block an seinen ursprünglichen Bildschirmkoordinaten neu gezeichnet.

Beachten Sie dabei, daß alle Zeichenoperationen so zugeschnitten (clipped) werden, daß sie in den aktuellen Schirm passen. Hier wird bei der nächstliegenden 16-Pixel- Grenze begonnen.

Die Routine in **BEISPIEL 15.2** gibt Ihnen eine Demonstration der BLOCK-Befehle. Für erfahrene Programmierer haben wir ein paar zusätzliche Extras miteingebaut. Für die überwiegende Mehrheit aller Anwendungen braucht man sie nicht, sie sind eigentlich nur dazu geeignet, wenn Sie ganz abgehobene Spezialeffekte auf dem Bildschirm erzeugen möchten.

Planes enthält eine Bitmap, mit den Farben, die in Ihrem Block gezeichnet werden. Der Bildschirm des Amiga ist in Segmente unterteilt, die als "Bitplanes" bezeichnet werden. Jede dieser Planes enthält ein Bit für jeden Punkt auf dem Amiga- Bildschirm. Wenn die Amiga-Hardware diesen Punkt anzeigt, kombiniert sie zur Berechnung der erforderlichen Farbnummer die Bits jeder Plane. Jedes Bit in Planes stellt den Status einer Bitplane dar. Wenn das Bit auf eins gestellt ist, wird die gewählte Plane von der Anweisung gezeichnet, sonst wird sie einfach völlig ignoriert. Die erste Plane wird dabei durch Bit Null dargestellt, die zweite durch Bit eins usw.

Normalerweise wird der Block in allen verfügbaren Bitplanes dargestellt. Das entspricht dem Bitmuster %111111.

Minterm bestimmt den Blitter-Modus, durch den Ihr Block auf den Bildschirm kopiert wird. Eine vollständige Beschreibung der verschiedenen Zeichen-Modi finden Sie im Abschnitt über den Befehl SCREEN COPY.

Am besten lernen Sie diese Optionen kennen, wenn Sie etwas mit ihnen experimentieren.

DEL BLOCK *(Löscht einen Block)*

DEL BLOCK n

Löscht einen oder mehrere Blöcke und führt den verwendeten Speicher wieder AMOS-Basic zu.

DEL BLOCK	Löscht **alle** aktuellen Blöcke.
DEL BLOCK n	Löscht den Block mit der Nummer *n*.

GET CBLOCK *(Speichert und packt ein Bild)*

GET CBLOCK n,x,y,sx,sy

Mit dem Befehl GET CBLOCK können Sie einen rechteckigen Bereich auf dem Bildschirm speichern und packen. Das Pack-System, das für diesen Befehl eingesetzt wird, wurde vor allem in Bezug auf die Geschwindigkeit optimiert. Deshalb ist es keineswegs so effizient wie die speziellen AMOS Pack-Routinen, die bei den Anweisungen PACK oder SPACK eingesetzt werden.

CBLOCK wird häufig dafür verwendet, den Bereich unter einer Dialogbox zu erfassen. Wenn der Dialog dann abgeschlossen ist, kann der Bildschirm schnell wieder in seinen Originalzustand versetzt werden. Sehen Sie sich dazu auch **BEISPIEL 15.3** im MANUAL-Unterverzeichnis an.

n stellt die Nummer Ihres Blockes dar und liegt zwischen 1 und 65535.

x,y sind die Koordinaten der linken oberen Ecke des Blockes. Die x-Koordinate wird dabei auf das nächstliegende Vielfache von 8 gerundet.

w,h geben die Abmessungen des Bereichs an, der gespeichert werden soll. Die Breite Ihres Blockes wird dabei stets genau auf ein Vielfaches von 8 gerundet.

PUT CBLOCK *(Zeigt einen Block an, der mit CBLOCK erzeugt wurde)*

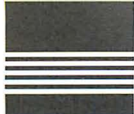
PUT CBLOCK n [,x,y]

Positioniert den Block mit der Nummer *n* auf dem aktuellen Schirm an den Koordinaten *x,y*. Wenn die Zielkoordinaten weggelassen werden, dann wird der Block an seiner ursprünglichen Bildschirmposition neu gezeichnet. Beachten Sie auch, daß x automatisch auf die nächstliegende 8-Pixel-Grenze gerundet wird.

DEL CBLOCK *(Löscht einen durch GET BLOCK definierten Block)*

DEL CBLOCK [n]

Löscht alle Blöcke - oder wenn ein Wert für *n* eingegeben wird, nur diesen Block - aus dem Speicher.



16: Menüs

Wenn Sie schon seit einiger Zeit mit dem Amiga arbeiten, dann sind Sie mit dem Konzept der Menüs bereits vertraut. AMOS hat auch hier das Unmögliche vollbracht und das bestehende System fast bis zur Unkenntlichkeit verbessert.

Sie können Menüs auf bis zu acht verschiedenen Ebenen erstellen, und jeder Menüpunkt kann beliebig auf dem Bildschirm angeordnet werden. Die Menütitel können in jeder Kombination von Farben oder Stilen gedruckt werden. Durch eine ganz erstaunliche Sprache zur Menüdefinition können Sie BOBs und Icons direkt in Ihre Menüs einbauen.

Aber nicht nur die Erstellung, auch die Abfrage der Menüs ist beeindruckend. Es gibt einen eingebauten, interruptgesteuerten ON MENU Befehl, der je nachdem, welche Option gewählt wird, zu einem bestimmten Punkt in Ihrem Programm springt. Darüber hinaus ist jede Menüoption durch die Anweisung MENU KEY direkt über die Tastatur zugänglich.

Was für phantastische Effekte Sie mit diesem System erzielen können, sehen Sie, wenn Sie **BEISPIEL 16.1** aus dem MANUAL-Unterverzeichnis laden. Erst wenn Sie sie in Aktion gesehen haben, können Sie wirklich glauben, wie unglaublich stark diese Befehle sind!

Wie man mit einem Menü arbeitet

Alle AMOS-Menüs werden durch Drücken der rechten Maustaste in der üblichen Weise aufgerufen. Nachdem Sie ein Menü aktiviert haben, können Sie eine Option direkt mit dem Maus-Cursor anwählen. Wenn Sie die Taste dann loslassen, erscheint die Nummer der Option, die Sie gewählt haben, in Ihrem Programm.

Menüs können dadurch neu angeordnet werden, daß Sie den Maus-Cursor auf die linke obere Ecke eines Punktes stellen und dann die **linke** Taste gedrückt halten. Nun erscheint ein kleiner Kasten auf dem Menübalken, den Sie dann mit der Maus über den Bildschirm ziehen können.

Außerdem wird ein Menü an seinem Platz festgehalten, wenn Sie die SHIFT-(Umschalt-)Taste gedrückt halten. Auf diese Art können Sie sich ein Menü in Ruhe ansehen, ohne eine der verschiedenen Optionen zu wählen. Darüber hinaus können Sie auch alle Mausfunktionen wie die Verlangsamung oder die Abfrage der Achsen in Verbindung mit Ihren Menüs einsetzen.

Wie man ein einfaches Menü erzeugt

Sie können Ihre Menüs entweder direkt aus Ihren Programmen heraus oder mittels eines speziellen Zusatzprogrammes, dem Menu Definer auf der AMOS- Programmdiskette, erstellen.

Wenn Sie zuvor noch nie mit Menüs gearbeitet haben, dann sind Sie vielleicht durch die Vielzahl der verfügbaren Menübefehle etwas beunruhigt. Wir wollen Ihnen nun einige der Grundfunktionen kurz beschreiben, um Sie ganz kurz und schmerzlos in die AMOS-Menüs einzuführen.

Das Festlegen der Titelzeile

Die Definition der Titelzeile *(title line)* stellt die erste Stufe in der Erstellung eines Menüs dar. Dazu verwenden Sie den Befehl MENU$. Die einfachste Version dieses Befehls hat das folgende Format:

MENU$ *(Legt einen Menütitel fest)*

MENU$(t,option$)

MENU$ erzeugt eine Titelzeile für Ihr Menü. Jeder Überschrift wird eine eigene Nummer - angefangen bei eins - zugewiesen, wobei die Nummern von links nach rechts steigen. Also wird der Titel ganz links durch eine Eins dargestellt, der Titel daneben durch eine Zwei usw.

Der Text in der Zeichenkette *Titel$* enthält die Bezeichnung der Option, die in Ihrem neuen Menü angezeigt wird. Hier ist ein einfaches Beispiel, in dem eine Menüzeile erzeugt wird, die nur aus zwei Titeln besteht, aus AKTION und MAUS.

```
Menu$ (1)=" Aktion "
Menu$ (2)=" Maus "
```

Beachten Sie hier den Leerschritt nach *Aktion*, er soll diesen Punkt von *Maus*, dem nächsten Menüpunkt, abtrennen. Jetzt müssen Sie eine Reihe von Optionen angeben, die jedem ihrer neuen Oberbegriffe zugeordnet werden. Diese Unterpunkte bilden dann einen vertikalen Balken, der sichtbar wird, wenn einer der Menüpunkte mit der Maus angewählt wird.

MENU$(t,option$) *(Angabe der Menüoptionen)*

Mit dieser zweiten Version von MENU$ können Sie eine Reihe von Optionen definieren, die auf dem Menübalken dann angezeigt werden.

t stellt die Nummer Ihres Menüpunktes dar, unter dem die Option erscheinen soll. o ist die Nummer der Option, die Sie in diesem Menübalken unterbringen möchten. Alle Optionen werden - begonnen mit eins - von oben nach unten durchnumeriert.

Die einzige Begrenzung für den Umfang Ihres Menüs liegt im verfügbaren Speicherplatz, aber es ratsam, sich bei jedem Menüpunkt auf höchstens 10 Optionen zu beschränken. Auf diese Art bleiben Ihre Menüs überschaubar.

Option$ enthält die Bezeichnung Ihrer neuen Option. Sie kann aus jedem beliebigen Text bestehen. Fügen Sie doch einmal die folgenden Zeilen zu dem obigen Beispiel hinzu:

```
Rem Aktion-Menü
Menu$ (1,1)=" Ende "
Rem Maus-Menü
Menu$ (2,1)=" Pfeil "
```

```
Menu$ (2,2)=" Zeiger "
Menu$ (2,3)= " Uhr "
Wait Key
```

Hier finden Sie eine Reihe von Alternativen für die AKTION- und MAUS-Menüs. Wenn Sie versuchen, dieses Programm so wie es jetzt ist, ablaufen zu lassen, geschieht überhaupt nichts. Das liegt daran, daß Sie die Menüs erst durch Aufrufen den Befehls MENU ON initialisieren müssen. Geben Sie diesen Befehl jetzt im vorstehenden Beispiel vor der Anweisung "Wait Key" ein. Nun können Sie das Beispiel ablaufen lassen und die Menüpunkte mit der Maus auswählen. Vergessen Sie aber nicht, zuerst die RECHTE Maustaste gedrückt zu halten!

MENU ON *(Aktiviert das Menü)*

MENU ON

MENU ON aktiviert ein Menü, das durch den MENU$-Befehl definiert wurde. Die Menüzeile erscheint jetzt automatisch, wenn der Anwender die RECHTE Maustaste drückt. Um das obengenannte Menü zu starten, fügen Sie die folgende Zeile nach den Definitions-Anweisungen ein.

```
Menu On
```

Gehen Sie dann ins Direktfenster und spielen Sie ein bißchen mit den Menüs. Wählen Sie die Optionen durch Drücken der rechten Maustaste.

Wie man ein einfaches Menü abfragen kann

Nachdem Sie Ihr Menü erstellt und das AMOS Menüsystem aktiviert haben, möchten Sie jetzt sicher wissen, wie Sie feststellen können, welche Optionen der Anwender gewählt hat. Dies können Sie durch eine einfache Version des CHOICE- Befehls erreichen.

=CHOICE *(Abfragen eines Menüs)*

Wahl=CHOICE

CHOICE weist den Wert -1 (richtig) aus, wenn der Anwender das Menü angewählt hat, sonst erscheint 0 (falsch). Diese Funktion wird nach jeder Abfrage automatisch auf 0 zurückgesetzt. Durch eine zweite Version dieser Anweisung kann man auch die Nummer des Titels, der gewählt wurde, ermitteln.

Überschrift=CHOICE(1)

Überschrift enthält jetzt die Nummer des Titels, der vom Anwender angewählt wurde. Über den Parameter 2 können Sie auch die Nummer der tatsächlich gewählten Option abfragen.

```
Punkt=CHOICE(2)
```

Versuchen Sie nun, zu dem oben dargestellten Beispiel die folgenden Zeilen hinzuzufügen:

```
Do
  If Choice and Choice(1)=1 Then Exit
  If Choice(1)=2 and Choice(2)<>0 Then Change Mouse Choice(2)
Loop
```

Hier wird die Form des Maus-Cursors verändert, je nachdem, welche Option Sie in dem Menü gewählt haben. In **BEISPIEL 16.2** im MANUAL-Unterverzeichnis finden Sie eine vollständige Demonstration dieser Menüs.

Menüfunktionen für Fortgeschrittene

Nun wollen wir uns einige der komplexeren Menüfunktionen in AMOS-Basic ansehen. Wenn Sie sie entsprechend einsetzen, können diese AMOS-Menüs Ihren Programmen ganz neue Dimensionen geben.

MENU$ *(Erzeugt ein Menü)*

MENU$(,,)=Normal$[,Wahl$][,Inaktiv$][,Hintergrund$]

MENU$ definiert die Darstellung jedes einzelnen Punktes in Ihren Menüs. Im Unterschied zu den normalen Amiga-Menüs müssen Sie sich bei diesen Menüpunkten nicht auf normalen Text beschränken. Sie können auch *eingebettete* Befehle enthalten, mit denen Sie an jeder beliebigen Stelle in der Menüzeile BOBs, Icons oder Grafiken zeichnen können.

Sie können hier auch jeden der Parameter in dieser Anweisung weglassen, und so Teile der Menübeschreibung unabhängig von einander verändern. Wenn Sie den Wert "" in Ihrer Menüzeichenkette einsetzen, so wird die vorhandene Einstellung dieses Parameters **gelöscht**. Den ursprünglichen Wert können Sie beibehalten, indem Sie ein Komma an der entsprechenden Stelle setzen. Dazu folgendes Beispiel:

```
Menu$(1)=" Action ","" : Rem Löscht zweite Option
Menu$(2)=" Mouse 2 "",, : Rem Verändert nur den Titel, sonst nichts
```

Die Stellung des Menüpunktes im eigentlichen Menü wird durch eine Reihe von bis zu acht Parametern angegeben, die voneinander durch Kommas abgetrennt sind. Das Format sieht im allgemeinen so aus:

(Menüpunkt)/(Menüpunkt,Option)/(Menüpunkt,Option,Unterpunkt)...

Normal$ ist eine Zeichenkette, die das normale Erscheinungsbild eines Menüpunktes in der Menüzeile bestimmt.

Wahl$ verändert die Wirkung, die das Hervorheben eines Menüpunktes mit der Maus hat. Als Default werden ausgewählte Menüpunkte in inverser Schrift dargestellt.

Inaktiv$ verändert das Erscheinungsbild eines Menüpunktes, wenn er durch den Befehl MENU INACTIVE deaktiviert wurde. Wenn Sie diese Zeichenkette weglassen, dann werden alle nicht aktiven Menüpunkte *kursiv* dargestellt.

Hintergrund$ erstellt einen Hintergrund für Ihre Menüpunkte, wenn sie das erste Mal gezeichnet werden. Im allgemeinen handelt es sich dabei um einen Kasten, der durch die internen Bar- oder Line-Befehle erstellt wird.

Von nun an wollen wir diese Parameter durch eine Standardangabe abkürzen:

```
Einstellung$=[,Wahl$][,Inaktiv$][,Hintergrund$]
```

Die Menüstruktur

Die Stellung, die ein Punkt in dem Menü innehat, wird durch seine Position in der *Menüstruktur* bestimmt.

```
Menu$(1)="Titel"
Menu$(1,1)="Option 1"
Menu$(1,2)="Option 2"
Menu$(1,2,1)="Punkt 1"
```

Auf diese Art wird ein einfaches Menü definiert. Die Struktur eines Menüs ähnelt der eines Feldes. Jede Ebene des Menüs wird durch eine bestimmte Dimension in dem Feld dargestellt, und wird durch eine besondere Version des MENU$-Befehls gesteuert.

Die erste Ebene stellt die *Titelzeile* dar, die oben an Ihren Menüs erscheint. Man kann Sie zum Beispiel mit dem folgenden Befehl erzeugen:

```
Menu$(n)=Titel$[Einstellung$]
```

n entspricht jetzt der Position des Titels gemessen vom linken Rand des Bildschirms, und *Einstellung$* entspricht den drei Zeichenketten, die Sie wahlweise eingeben können, um das generelle Erscheinungsbild des Menüs festzulegen. Dabei ist es wichtig, erst den Titel Ihres Menüs einzugeben, da dieser die **Dimensionen** des Arrays bestimmt. Alle anderen Menüpunkte können dann in ganz beliebiger Reihenfolge angeordnet werden.

Jeder Titel ist mit einer Reihe von Menüoptionen verbunden, die erscheinen, wenn das Menü gewählt wird. Sie stellen die zweite Ebene der Menüstruktur dar und werden durch die zweite Version des MENU$-Befehls erstellt.

```
Menu$(n,Option)=Punkt$[Einstellung$]
```

Option enthält die Nummer des Menüpunktes, dabei wird mit dem Zählen links oben am Menübalken begonnen. Die Anzahl der Optionen, die einem Titel zugeordnet sind, ist *eigentlich* nur durch den verfügbaren Speicherplatz beschränkt.

Jeder Option können nun wiederum eigene Untermenüs zugeordnet werden, dabei können im Ganzen bis zu acht verschiedene Ebenen erzeugt werden.

Menu$(n,Option,Unterpunkt)=Punkt$[Einstellung$]

Nachdem Sie das Menü erstellt haben, können Sie es jederzeit erweitern oder verändern. Während Sie jedoch ein Menü erzeugen, dürfen Sie den aktuellen Schirm nicht verändern, sonst erhalten Sie eine Fehlermeldung.

Am besten machen Sie sich mit der Menüstruktur vertraut, indem Sie sich das Programm **BEISPIEL 16.3** im MANUAL-Unterverzeichnis ansehen.

=CHOICE *(Abfragen eines Menüs)*

Punkt=CHOICE[(Dimension)]

Mit der CHOICE-Funktion können Sie überprüfen, ob eine Option des aktuellen Menüs hervorgehoben wurde. Wenn ein Menüpunkt gewählt wurde (von der obersten bis zur untersten Ebene), dann weist CHOICE den Wert -1 (richtig) aus, sonst erscheint 0 (falsch). Nachdem Sie diese Funktion aufgerufen haben, wird der Status des Menüs automatisch auf 0 (falsch) zurücksetzt. Auf diese Art wird vermieden, daß das einmalige Aufrufen eines Menüpunktes versehentlich mehrmals erfaßt wird.

Eine zweite Version des CHOICE-Befehls dient dazu, die gewählte Option auf der entsprechende Ebene zu ermitteln.

Punkt=CHOICE(Dimension)

Dimension gibt die Menüebene an, die abgefragt werden soll. Wie bereits erwähnt, entspricht hier der Wert 1 der Titelzeile des Menüs. Entsprechend stellen die Zahlen 2 bis 8 die jeweilige Option dar, die gewählt wurde. Wenn ein Menüpunkt nicht gewählt wurde, dann wird *Punkt* mit dem Wert Null geladen. Hier ist ein Beispiel:

```
Menu$(1)="Menu"
Menu$$(1,1)="Option 1"
Menu(1,2)="Option 2"
Menu(1,2,1)="Option 2.1"
Menu On
Do
  If Choice Then Print Choice(1),Choice(2),Choice(3)
Loop
```

Wenn Sie dieses System auf umfangreichere Menüs anwenden möchten, müssen Sie eine lange Liste von IF...THEN Anweisungen einsetzen, um alle verschiedenen

Möglichkeiten abzudecken. Das würde zu einer geringer, aber doch entscheidenden Verzögerung in Ihrem Programm führen, wenn die Menüs abgefragt werden. Außerdem wäre es dann sehr schwierig, Ihr Programm später zu verändern.

Glücklicherweise hat auch hier AMOS-Basic eine Methode parat, mit der Sie auch die größten Menüs ohne viel Federlesens in den Griff bekommen.

ON MENU PROC *(Automatische Menüwahl)*

ON MENU PROC Proc1,[,Proc2...]

Sie können jedem Titel in Ihrem Menü eine eigene Prozedur zuweisen, die automatisch ausgeführt wird, sobald der Anwender eine Option wählt. Dieser Befehl entspricht im wesentlichen den folgenden Zeilen im AMOS Basic-Code:

```
If CHOICE
   If CHOICE(1)=1
            Proc1
   Endif
   If CHOICE(1)=2
            Proc2
   Endif
:     :     :
:     :     :
Endif
```

Es besteht jedoch ein entscheidender Unterschied zwischen dem Befehl ON MENU und den obengenannten Anweisungen. ON MENU wird 50mal pro Sekunde durch Interrupts ausgeführt und hat keine Auswirkungen auf den Ablauf Ihres Programms. Das bedeutet, daß Ihr Programm etwas völlig anderes tun kann, während die Menüs durch dieses System abgefragt werden.

Wenn der Anwender einen Menüpunkt wählt, dann wird die entsprechende Prozedur sofort ausgeführt, ohne daß von seiten der Programms irgendeine Reaktion erfolgt. Ihre Prozedur kann dann über den CHOICE-Befehl abfragen, welche Option gewählt wurde, und die entsprechenden Schritte durchführen.

Nachdem die Prozedur abgeschlossen ist, kehrt Ihr Programm zu der Anweisung zurück, die auf den Befehl ON MENU folgt. Hier ist ein Beispiel dazu:

```
Menu$(1)="Aktion" : Menu$(1,1)="Zähle" : Menu$(1,1)="Ende"
Menu on: Rem Activiert Menü
On Menu Proc AKTION
On Menu On : Rem Activiert Befehl On Menu
Do : Rem Tipp ein paar Zeichen ein
   X$=Inkey$ : If X$<>""Then Print X$; : Inc W
Loop
Procedure AKTION
```

```
  Shared W
  If Choice(2)=1
     Locate 0,0:Print "Du hast ";W;" Buchstaben getippt":W=0
     On Menu On : Rem Initialisiert Menüs
  Endif
  If CHOICE(2)=2 Then Edit
End Proc
```

Zu diesem Beispiel sind ein paar wichtige Dinge zu sagen. Erstens beachten Sie bitte, wie wir die MENU-Sequenz durch den Befehl ON MENU ON aktivieren. Dieser Befehl **muß** aufgerufen werden, nachdem die Menüprozedur abgeschlossen ist, denn das Menüsystem muß dann neu gestartet werden. Beachten Sie auch, daß wir anstatt INPUT die Anweisung INKEY$ eingesetzt haben. Der Befehl INPUT hält nämlich die Menüabfrage an, während Sie eine Zeile eingeben. Alle anderen Befehle, einschließlich WAIT, WAIT VLB und WAIT KEY können ganz problemlos angewandt werden. Ein weiteres Beispiel hierzu finden Sie in **BEISPIEL 16.4**.

ON MENU GOSUB *(Automatische Menüwahl)*

ON MENU GOSUB Label1 [,Label2,...]

ON MENU GOSUB gibt eine bestimmte Subroutine aus eine Reihe von Subroutinen ein, je nachdem, welche Option der Anwender gewählt hat. Nachdem Sie diesen Befehl aufgerufen und Ihre Subroutinen erstellt haben, werden die Menüs automatisch alle 50stel Sekunden abgefragt.

Beachten Sie, daß jeder Titel in der Menüzeile von seiner eigenen Subroutine verwaltet wird. Das unterscheidet sich völlig vom AMIGA Basic, bei dem das gesamte Menü nur durch eine einzige Subroutine gesteuert wird.

Nachdem Sie diesen Befehl eingesetzt haben, sollten Sie das Menüsystem durch Aufrufen der Anweisung ON MENU aktivieren. Die Menüs müssen auf diese Weise neu initialisiert werden, bevor Sie mit RETURN in Ihr Hauptprogramm zurückspringen. Beachten Sie auch, daß *Label* **nicht** durch einen Ausdruck ersetzt werden kann, da die Sprungmarke nur einmal berechnet wird, während das Programm abläuft. Siehe dazu auch die Befehle ON MENU PROC und ON MENU ON/OFF.

ON MENU GOTO *(Automatische Menüwahl)*

ON MENU GOTO Label1[,Label2,...]

Dieser Befehl wurde jetzt durch die Anweisungen ON MENU PROC und ON MENU GOSUB ersetzt, die eine größere Leistung bieten. ON MENU GOTO wurde in AMOS-Basic in erster Linie miteinbezogen, um Kompatibilität zu Programmen, die in STOS Basic geschrieben sind, zu bieten. Wenn ein Menü gewählt wird, dann springt das Programm zur entsprechenden Sprungmarke. Siehe auch ON MENU PROC und ON MENU GOSUB.

ON MENU ON/OFF

(Aktiviert/deaktiviert die automatische Menüwahl)

ON MENU ON

ON MENU ON aktiviert das automatische Menüsystem, das durch die Befehle ON MENU PROC/GOSUB/GOTO erzeugt wurde. Nachdem eine Subroutine auf diese Art angesprochen wurde, wird das System **außer Kraft gesetzt**. Deshalb müssen Sie unbedingt das System durch Aufrufen von ON MENU ON wieder aktivieren, bevor Sie in Ihr Hauptprogramm zurückgehen.

ON MENU OFF

Diese Anweisung unterbricht das automatische Menüsystem zeitweise. Das ist nützlich, wenn Ihr Programm eine Prozedur ausführt, die ohne Unterbrechungen durchgeführt werden muß, wie zum Beispiel das Laden oder Speichern von Informationen auf Diskette. Die Menüs können dann durch ON MENU ON wieder aktiviert werden.

ON MENU DEL

(Löscht die Sprungmarken, die von ON MENU verwendet wurden)

ON MENU DEL

Dieser Befehl löscht die interne Auflistung der Sprungmarken oder Prozeduren, die durch die ON MENU Anweisungen erzeugt wurden. Sie können jetzt Ihre Menüs einem anderen Teil Ihres Programms zuordnen, indem Sie wieder ON MENU aufrufen. **Achtung!** Setzen Sie diesen Befehl erst ein, **nachdem** Sie die Menüs mit ON MENU OFF deaktiviert haben.

Abkürzungstasten

Trotz der unbestreitbaren Vorteile der Menüs ziehen es einige Anwender vielleicht vor, die Optionen eines Programms direkt über die Tastatur einzugeben. Obwohl die Auswahl durch Menüs für Anfänger bestimmt leichter ist, kann es viel schneller gehen, wenn Sie die Option über die Tastatur eingeben, vor allem, wenn Sie mit dem Programm sehr gut vertraut sind.

AMOS-Basic ermöglicht es Ihnen, jeder Ihrer Menüoptionen eine Abkürzungstaste zuzuweisen. Dieser Tastendruck wird dann genauso interpretiert, als hätte der Anwender die entsprechende Option über die Menüzeile ausgewählt. Dieses System kann bei jedem der Menübefehle in AMOS-Basic eingesetzt werden, einschließlich von ON MENU.

MENU KEY *(Weist einem Menüpunkt eine Taste zu)*

MENU KEY (,,) TO c$
MENU KEY (,,) TO Scan[,shift]

Hier können Sie jedem Menüpunkt eines zuvor definierten Menüs eine bestimmte Taste zuweisen. Die einzige Beschränkung besteht darin, daß sich der Punkt, den Sie angeben, auf der untersten Ebene Ihres Menüs befinden muß. Sie können also mit einer Abkürzungstaste kein Untermenü aufrufen, da jeder Befehl einer bestimmten Option in Ihrem Menü entsprechen muß.

c$ ist eine Zeichenkette, die ein bestimmtes Zeichen enthält, das der Menüoption zugewiesen wird. Alle weiteren Zeichen in dieser Kette werden ignoriert.

Jeder Taste auf der Tastatur des Amiga wird ein bestimmter *Scan-Code* zugewiesen. Durch die Verwendung dieses Codes können Sie einem Menü Tasten zuweisen, die keine ASCII-Entsprechung haben. Hier ist eine Reihe von Scan- Codes, die Sie in Ihrem Menüs einsetzen können:

Scan-Code	Tasten
80-89	Funktionstasten F1-F10
95	Hilfe
69	Esc

Shift ist eine Bitmap, über die Sie wahlweise solche Kombinationen der Steuertasten wie Alt+Hilfe oder Ctrl+D abfragen können. Shift hat folgendes Format:

Bit	Abgefragte Taste	Bemerkungen
0	Linke Shift-Taste	Es kann immer nur jeweils eine der Shift-Tasten abgefragt werden
1	Rechte Shift-Taste	
2	Caps Lock	Entweder AN oder AUS
3	Ctrl	
4	Alt links	
5	Alt rechts	
6	Amiga links	Bei manchen Tastaturen ist das die Commodore-Taste
7	Amiga rechts	

Wenn Sie in diesem Bitmuster mehr als ein Bit einstellen, müssen Sie aber auch zum Aufrufen Ihres Menüpunktes mehrere Tasten gleichzeitig drücken. Alle Abkürzungstasten können außer Kraft gesetzt werden, indem Sie die Anweisung MENU KEY ohne Angabe von Parametern eingeben. Hier ist ein Beispiel:

```
Menu Key(1,10) : Rem Schaltet Abkürzungstaste für Punkt (1,10) ab
```

Mit Hilfe des Befehl MENU KEY ist das Hinzufügen von Abkürzungstasten zu einem Menü eine Kleinigkeit, deshalb sollten Sie sie immer in Ihre Programme einbauen. Hier ist ein Beispiel, mit dem die 10 Funktionstasten des Amiga abgefragt werden:

```
Menu$(1)="Funktionstasten"
For A=1 To 10
   OPT$=" F"+Str$(A)+""
   Menu$(1,A)=OPT$
   Menu Key(1,A) To 79+A
Next A
Menu On
Do
   If Choice Then Print "Du hast die Funktionstaste gedrückt";Choice(2)
Loop
```

Befehle zur Menüsteuerung

MENU ON *(Aktiviert ein Menü)*

MENU ON [Bank]

Durch MENU ON aktivieren Sie ein Menü, das Sie zuvor in Ihrem Programm definiert haben. Dieses Menü erscheint, wenn der Anwender die rechte Maustaste das nächste Mal drückt, und die Optionen werden wie üblich angezeigt. Wenn Sie bei dieser Anweisung auch eine *Bank*nummer angeben, dann wird das Menü der entsprechenden Speicherbank entnommen. Weitere Einzelheiten dazu finden Sie unter MAKE MENU BANK.

MENU OFF *(Setzt ein Menü zeitweise außer Kraft)*

MENU OFF

Dies ist das genaue Gegenteil der Anweisung MENU ON. Hier wird das gesamte Menü zeitweise außer Kraft gesetzt. Sie können das Menü dann jederzeit durch Aufrufen von MENU ON wieder starten.

MENU DEL *(Löscht einen oder mehrere Menüpunkte)*

MENU DEL löscht das gewählte Menü aus dem Speicher des Amiga und macht den freigewordenen Platz wieder für den Rest Ihres Programmes zugänglich. Für diese Anweisung gibt es zwei mögliche Ausführungen:

MENU DEL

Löscht das **gesamte** Menü. **Achtung!** Dieser Befehl kann nicht revidiert werden!

MENU DEL (,,)

Löscht nur einen Teil des Menüs. Die Parameter (,,) enthalten eine Reihe von bis acht Werten, die jeweils durch Komma abgetrennt sind. Sie geben die genaue Position des jeweiligen Menüpunktes in der Menüstruktur an. Hier ist ein Beispiel:

```
Menu Del(1) : Rem Löscht Titel Nummer 1
Menu Del(1,2) : Rem Löscht Option 2 von Titel 1
```

MENU TO BANK

(Speichert die Menüdefinitionen in einer Speicherbank)

MENU TO BANK n

Mit dieser Anweisung können Sie einen ganzen Menübaum in der Speicherbank mit der Nummer *n* speichern. Wenn die Bank *n* bereits existiert, erhalten Sie die Fehlermeldung *Bank schon reserviert.*

Nachdem Sie ein Menü auf diese Art gespeichert haben, wird es automatisch zusammen mit Ihrem Basic-Programm gespeichert. Sie können den Umfang Ihrer Programm-Listings ganz beträchtlich reduzieren, indem Sie Ihre Menüdefinitionen in einer Speicherbank unterbringen. Dadurch wird nämlich wertvoller Platz in Editorspeicher frei, und Sie können längere Basic-Programme schreiben, ohne auch gleichzeitig mehr Speicher zu benötigen.

BANK TO MENU

(Gibt die Menüdefinition an, die in einer Speicherbank gespeichert ist)

BANK TO MENU n

Durch diesen Befehl wird aus den Menüdaten, die in der Bank mit der Nummer *n* gespeichert sind, eine Menüdefinition erstellt.

Ihr Menü wird genau in denselben Zustand zurückversetzt, in dem es ursprünglich gespeichert wurde. Wenn es sich hier um ein komplexes Menü handelt, dann kann die Ausführung dieses Befehls etwas Zeit in Anspruch nehmen. Sie können Ihr neues Menü dann mit der Anweisung MENU ON aktivieren.

MENU CALC

(Neuberechnen eines Menüs)

MENU CALC

Eine der angenehmsten Eigenschaften der AMOS Menüs ist es, daß man Sie leicht im Laufe eines Programmes verändern kann. Nachdem Sie Ihre erste Definition einmal erstellt haben, können Sie nach Belieben neue Punkte hinzufügen und die vorhandenen Optionen ersetzen.

Alle Ihre Menüpunkte werden immer automatisch neu positioniert, wenn das Menü mit der rechten Maustaste aufgerufen wird. Wenn Ihre Menüs sehr umfangreich sind, dauert das natürlich ein Weilchen. Durch MENU CALC können Sie diesen Vorgang an der geeignetsten Stelle in Ihrem Programm durchführen, und so unnötige und störende Verzögerungen vermeiden.

Damit der Anwender das Menü nicht aufrufen kann, während es geändert wird, sollten Sie die Menüs am besten durch das Einsetzen MENU OFF am Anfang Ihrer Prozedur zeitweise außer Kraft setzen. Wenn Sie dann fertig sind, können die Menüs mit MENU ON wieder gestartet werden.

Menüs, die sich mit dem Spielverlauf verändern, eignen sich besonders gut für Abenteuerspiele. So können dann an jedem Standort spezielle Menüoptionen dem Verhalten des Spielers angepaßt werden.

Eingebettete Menübefehle

Jede Menüsteuerkette kann wahlweise auch eine Reihe von eingebetteten Befehlen umfassen, durch die Sie das Erscheinungsbild Ihrer Menüs unglaublich gut anpassen können. Diese Befehle werden in runde Klammern() gestellt, und die einzelnen Anweisungen voneinander durch Doppelpunkte abgetrennt. Hier ist ein Beispiel dazu:

```
Menu$(1)="(Locate 10,10 : Ink 1,1) Hallo"
```

Jede Anweisung besteht nur aus zwei Zeichen, die entweder klein- oder großgeschrieben werden können. Alles andere wird völlig ignoriert. Bei den meisten Befehlen müssen Sie dazu noch eine oder mehrere Zahlen eingeben. Diese Zahlen dürfen jedoch **keinen** mathematischen Ausdruck darstellen, da Ausdrücke hier nicht berechnet werden. Die betreffenden Befehle werden unten aufgeführt.

Achtung: Die beiden entscheidenden Zeichen, die den Befehl bezeichnen, werden im folgenden groß- und fettgedruckt dargestellt.

BOB *(Zeichnet ein BOB)*

BOb n

Der Befehl BOb zeichnet das BOB mit der Nummer *n* an der aktuellen Position. Dabei wird der Hot Spot des BOBs nicht berücksichtigt. Alle Koordinaten werden in Bezug zur linken oberen Ecke angegeben. Außerdem wird die Farbe Null normalerweise transparent dargestellt. Das können Sie durch den AMOS Basic- Befehl NOMASK ändern. Hier ist ein Beispiel dazu:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Menu$(1)="(Bob 1) 1":Menu$(1,1)="(Bob 2) 2":Menu$(1,2)=" (Bob 3) 3"
Menu on : Wait Key
```

ICON *(Zeichnet ein Icon)*

ICon n

ICon zeichnet das Icon mit der Nummer n an der aktuellen Cursor-Position. Anders als bei den BOBs ist die Farbe Null hier normalerweise **nicht** transparent. Weitere Einzelheiten hierzu finden Sie bei dem Basic-Befehl MAKE ICON MASK.

LOCATE *(Bewegt den Grafik-Cursor)*

LOcate x,y

Der Befehl LOcate bewegt den Grafik-Cursor zu den Koordinaten x,y. Diese Koordinaten werden im Verhältnis zu der linken oberen Ecke der Menüzeile erfaßt. Dabei ist zu beachten, daß der Grafik-Cursor nach jeder Anweisung stets unten rechts an dem Objekt, das gerade gezeichnet wurde, positioniert wird. Durch diese Koordinaten können Sie auch feststellen, wo weitere Punkte Ihres Menüs positioniert sind. Das geht folgendermaßen:

```
Menu$(1)="Beispiel":Menu$(1,1)="Locate (Lo 50,50) in Aktion "
Menu$(1,2)="Kannst Du meine Koordinaten raten"
Menu on : Wait Key
```

INK *(Bestimmt Farben für Zeichnung und Hintergrund)*

INk n,Wert

Durch den INk-Befehl erfolgt eine Zuweisung der Farbindizes für die Darstellung von Zeichnung, Hintergrund und Umrandung (PEN, PAPER und OUTLINE). Hier sind die verschiedenen Möglichkeiten aufgeführt:

n	Wirkung
1	Bestimmt Farbe für Zeichnung/Text (PEN)
2	Bestimmt Farbe für Hintergrund (PAPER)
3	Bestimmt Farbe für Umrandung (OUTLINE)

SFONT *(Bestimmt Font)*

SFont n

SFont macht den **Grafik**font mit der Nummer n zum aktuellen Font. Dieser Font wird jetzt auch für alle folgenden Menüpunkte eingesetzt. Bevor diese Anweisung aber ausgeführt wird, **müssen** Sie GET FONTS aufrufen, sonst kann dieser Befehl nur auf die beiden ROM-Fonts zugreifen. Siehe dazu auch **BEISPIEL 16.5**.

SSTYLE *(Bestimmt Fontstil)*

SStyle n

Mit diesem Befehl setzen Sie den Stil des aktuellen Fonts auf *n*. *n* stellt ein Bitmuster im folgenden Format dar:

Bit	Wirkung
0	Setzen Sie dieses Bit auf 1 um Ihre Zeichen zu unterstreichen.
1	Wählt Fettdruck.
2	Aktiviert kursive Schrift.

LINE *(Zieht eine Linie)*

LIne x,y

Mit dem Befehl LIne wird von der aktuellen Cursor-Position bis zu den Koordinaten *x,y* eine Linie gezogen. Siehe dazu auch **BEISPIEL 16.6**.

SLINE *(Wählt den Stil der Linien)*

SLine p

SLine bestimmt den Stil, in dem alle folgenden, durch den LINE-Befehl erzeugten Linien dargestellt werden. Dieser Stil wird durch das Bitmuster in *p* festgelegt. Da Ausdrücke hier nicht berechnet werden, sollte man dieses Bitmuster vor dem Einsatz immer in einen Dezimalwert umrechnen. In **BEISPIEL 16.7** finden Sie eine einfache Demonstration der möglichen Stilarten.

BAR *(Zeichnet einen Balken)*

BAr x,y

Hier wird ein rechteckiger Balken gezeichnet, angefangen bei den aktuellen Cursor-Koordinaten *x,y*. **BEISPIEL 16.8** demonstriert Ihnen dies.

PATTERN *(Zeichnet ein Muster)*

PAttern n

Setzt das Füllmuster n ein, um die mit dem Befehl BAR erzeugten Balken zu füllen. Im AMOS MANUAL-Unterverzeichnis finden Sie dazu **BEISPIEL 11.8**.

OUTLINE *(Umrahmt einen Balken)*

OUtline Flag

OUtline umrahmt alle im folgenden erstellten Balken in der Farbe, die zur Zeit für die Umrandung (outline colour = ink 3) eingestellten Farbe. Der Wert eins aktiviert das Umranden und durch Null wird der Rahmen entfernt.

ELLIPSE *(Zeichnet eine Ellipse)*

ELlipse r1,r2

ELlipse zeichnet ein Oval mit dem Radiien *r1* und *r2* an den aktuellen Cursor-Koordinaten. Einen Kreis können Sie zeichnen, indem Sie einfach r1 mit r2 gleichsetzen. Siehe dazu auch **BEISPIEL 16.9**.

PROC *(Aufrufen einer Prozedur)*

PRoc NAME

Durch die PROC-Anweisung können Sie jede AMOS-Basic Prozedur direkt aus einer Menüzeile heraus aufrufen. Die aufgerufene Prozedur darf jedoch **keine** Parameter enthalten, sonst erhalten Sie einen Syntaxfehler.

Mit diesem Befehl können Sie das Menü ganz genau auf Ihre Anforderungen zuschneiden, ohne sich auf die verfügbaren Menübefehle beschränken zu müssen. Um

diese Funktion jedoch gut einsetzen zu können, sollten Sie die zugrundeliegende Theorie ein bißchen verstehen.

Am Anfang Ihrer Prozedur werden die folgenden Werte in den Registern des 68000-Prozessors gespeichert.

DREG(0) X-Coord

Dies ist die X-Koordinate der oberen linken Ecke des aktuellen Menüpunktes. Zeichnen Sie Ihre Grafiken nicht auf den links von diesem Punkt liegenden Teil des Bildschirms, da so das Neuzeichnen des Menüs gestört wird und irritierende Nebenwirkungen auftreten können.

DREG(1) Y-Coord

Das Register D1 enthält die Y-Koordinate Ihres Menüpunktes. Wie bei der X- Koordinate sollten Sie sich auch hier mit Ihren Zeichenoperationen auf den Bereich, der unterhalb dieses Punktes liegt, beschränken, um so Fehler zu vermeiden.

DREG(2) Status der Zeichenoperation

Dieses Register enthält den aktuellen Status der Menüoperationen. Wenn es den Wert 0 (falsch) aufweist, dann wird der Menüpunkt gerade gezeichnet. In diesem Fall müssen Sie DREG(0) und DREG(1) mit den Koordinaten der rechten unteren Ecke Ihrer Menüzone laden und sofort aus dieser Prozedur aussteigen.

Wenn DREG(0) -1 (richtig) lautet, dann können Sie die Grafikoperationen in Ihrer Prozedur durchführen. Wenn Sie damit fertig sind, sollten Sie die Koordinaten der rechten unteren Ecke Ihres Punktes wie zuvor in DREG(0) und DREG(1) ausweisen.

DREG(3) Status des Menüpunktes

D3 enthält den Wert -1, wenn das Menü hervorgehoben und die erste Menüzeichenkette angezeigt wird. Andernfalls erscheint der Wert 0.

DREG(4)

D4 wird auf RICHTIG (TRUE) gestellt, wenn dieser Menüzweig erstmals eröffnet wird.

AREG(1) Adresse der reservierten Zone

Dies ist die Adresse der durch RESERVE erzeugten Zone. Auf diese Art können verschiedene Prozeduren miteinander kommunizieren. Nähere Einzelheiten dazu finden Sie unter RESERVE.

Die allgemeine Struktur einer Menüprozedur lautet folgendermaßen:

```
Procedure ITEM
  If DREG(2)
        X=DREG(0):Y=DREG(1)
        ...Den Punkt zeichnen...
       Endif
       DREG(0)=BX:Rem X Koord rechte untere Ecke des Menüpunktes
       DREG(1)=BY:Rem Y Koord rechte untere Ecke des Punktes
  Endproc
```

Über die Koordinaten BX und BY werden die Dimensionen des Menüpunktes wie auf dem Bildschirm angezeigt festgelegt. Diese Koordinaten **müssen** in die Register D0 und D1 geladen werden, bevor Sie Ihre Prozedur verlassen, da sie zur Erstellung des Menübalkens erforderlich sind.

Aus Ihrer Prozedur heraus können Sie fast alle AMOS Anweisungen ausführen, und sogar auch andere Prozeduren. Aber manche Anweisungen sind ganz streng verboten! Wenn Sie diese Befehle einsetzen, erhalten Sie keine Fehlermeldung aber Ihr Amiga kann ganz unvermittelt abstürzen.

- Verändern Sie **niemals** den aktuellen Schirm aus einem Menü heraus. Das ist eines der sichersten Mittel um den Amiga zum Absturz zu bringen!
- Arbeiten Sie nicht mit SET oder RESET ZONE.
- Vermeiden Sie alle Anweisungen wie WAIT, WAIT KEY oder INPUT und INKEY$, die den Ablauf Ihres Programmes unterbrechen.
- Alle Diskettenoperationen sind streng verboten!
- Alle Korrektur-Routinen in Ihren Prozeduren werden völlig ignoriert. Wenn ein Fehler auftritt, wird das Menü geschlossen und Ihr Programm kehrt in den Editor zurück.

Wenn Sie ihn mit der gebotenen Vorsicht behandeln, können Sie durch den PROC-Befehl ganz atemberaubende Wirkungen erzielen. Laden Sie doch **BEISPIEL 16.10** aus dem MANUAL-Unterverzeichnis und sehen Sie sich ein Beispiel an. Außerdem finden Sie auch unter MENU CALLED weitere Angaben.

RESERVE *(Reserviert einen lokalen Datenbereich für eine Prozedur)*

REserve n

Reserviert einen Speicher von n Bytes für diesen Menüpunkt. Über die in AREG(1) enthaltene Adresse erhalten Sie Zugang zu diesem Bereich aus Ihrer Menüprozedur heraus. Alle Zeichenketten im aktuellen Menüobjekt teilen sich diesen Dateibereich, den Sie erzeugt haben. Hier kann zum Beispiel der Austausch von Parametern zwischen den verschiedenen Prozeduren, die ein Menüpunkt aufruft, stattfinden.

MENU CALLED *(Zeichnet einen Menüpunkt ständig neu)*

MENU CALLED(,,)

Der Befehl MENU CALLED zeichnet den gewählten Menüpunkt automatisch 50mal pro Sekunde neu. Normalerweise setzt man diese Anweisung in Verbindung mit einer Menüprozedur ein, um animierte Menüpunkte zu erzeugen, die sich vor Ihren Augen ständig verändern.

Diese Funktion können Sie erst einsetzen, wenn Sie zuvor anhand der oben aufgeführten Richtlinien eine Menüprozedur definiert haben. Dann können Sie diese Prozedur über die entsprechenden Titelketten durch einen eingebetteten PRoc- Befehl aufrufen. Zum Schluß aktivieren Sie dann den Aktualisierungsprozeß durch das Aufrufen von MENU CALLED. Wenn der Anwender den gewünschten Punkt anzeigt, wird Ihre Prozedur wiederholt vom Menüsystem angesprochen.

Da Ihre Prozedur 50mal pro Sekunde aufgerufen wird, sollte sie natürlich so schnell wie möglich zum Menü zurückspringen. Dadurch bleibt genug Zeit für die erfolgreiche Aktualisierung des übrigen Menüs.

Ihre eingebettete Prozedur kann Ihren Menüpunkt problemlos mit BOBs oder Sprites animieren. Da für die Menüpunkte jedoch kein Double-Buffering eingesetzt wird, können Ihre BOBs auf dem Bildschirm etwas flimmern. Deshalb ist es vielleicht besser, wenn Sie zu diesem Zweck berechnete Sprites einsetzen. Ein anderer Weg besteht darin, daß Sie Ihr Display mit den normalen AMOS Grafikbefehlen zeichnen. Ein Beispiel dafür finden Sie in **BEISPIEL 16.11** im MANUAL-Unterverzeichnis.

MENU ONCE *(Schaltet das automatische Neuzeichnen ab)*

MENU ONCE(,,)

MENU ONCE schaltet das automatische Aktualisierungssystem, das Sie mit dem Befehl MENU CALLED aktiviert haben, wieder ab. Von nun an wird jeder Menüpunkt jedesmal, wenn das Menü aufgerufen wird, nur einmal neu gezeichnet.

```
Menu Once(1,1)
```

Alternative Menü-Stile

Normalerweise werden die Titel einer Menüzeile in einer horizontalen Linie angezeigt und die Optionen sind darunter in einem vertikalen Balkenmenü angeordnet. Wenn Sie aber eine etwas interessantere oder kreativere Darstellungsmöglichkeit wählen möchten, so können Sie das Format jeder Ebene Ihres Menüs anhand der folgenden Anweisungen verändern:

MENU LINE *(Zeigt das Menü als horizontale Zeile von Menüpunkten an)*

MENU LINE Ebene
MENU LINE(,,)

Der Befehl MENU LINE zeigt die Menüoptionen der gewünschten Ebene in Form einer horizontalen Zeile an. Diese Menüzeile beginnt in der linken Ecke des ersten Titels und reicht zur rechten unteren Ecke des letzten Titels.
MENU LINE Ebene

Definiert den Menüstil einer ganzen Ebene Ihres Menüs. Diese Anweisung sollten Sie nur aufrufen, während Sie das Menü definieren.

MENU LINE(,,)

Normalerweise verwendet man nur die Version mit *level* dieses Befehls. Wenn Sie hier einzelne Menüpunkte auf Line oder Bar stellen, können Sie ganz bizarre Ergebnisse erhalten, aber wer weiß, wofür das mal gut sein kann!

MENU TLINE *(Zeigt ein Menü als ganze Zeile an)*

MENU TLINE Ebene
MENU TLINE(,,)

MENU TLINE zeigt einen Abschnitt des Menüs als ganze Zeile (*total line*) an, die sich von ganz links auf dem Bildschirm bis nach ganz rechts erstreckt. Dabei wird immer die ganze Zeile gezeichnet, selbst wenn sich der erste Menüpunkt in der Mitte des Bildschirms befindet.

Ebene ist eine Zahl zwischen 1 und 8, die den Teil des Menüs bestimmt, auf den diese Anweisung wirken soll. Dies ist die normalerweise übliche Version dieses Befehls. Sie sollten diese Anweisung stets dann aufrufen, wenn Sie Ihre Menüdefinition vornehmen, sonst hat sie keine Wirkung.

Mit der zweiten Version dieser Anweisung können Sie die Darstellung eines Menüs verändern, nachdem Sie es erstellt haben. Hier ist ein Beispiel dafür:

```
Menu Line(1,1) : Rem Zeigt Menü 1,1 als Zeile an
```

MENU BAR *(Zeigt einen Teil des Menüs als Balken an)*

MENU BAR Ebene
MENU BAR(,,)

Hier werden die gewählten Menüpunkte in Form eines vertikalen Balkens dargestellt. Die Breite dieses Balkens wird automatisch den Dimensionen des längsten Menüpunktes angeglichen.

Ebene ist eine Zahl, die angibt welcher Teil des aktuellen Menüs von dieser Anweisung betroffen ist. Als Default wird diese Option auf die Ebenen 2 bis 8 Ihres Menüs angewandt. Beachten Sie, daß Sie diese Ausführung der Anweisung MENU BAR nur während der Initialisierungsphase Ihres Programmes einsetzen können. Wenn Sie sie aufrufen, nachdem ein Menü bereits aktiviert wurde, hat sie keine Wirkung.

(,,) ist ein Reihe von Parametern, die es Ihnen ermöglichen, den Stil Ihres Menüs zu verändern, nachdem es erstellt wurde. Hier ist ein Beispiel für MENU BAR und MENU TLINE:

```
Flag=0
SET_MEN
Do
   If Choice and Choice(1)=2 and Choice(2)=1 Then ALTER
Loop
Procedure SET_MEN
   Menu$(1)=" Balkendemo " : Menu$(2)=" Wähle unten "
   Menu$(1,1)=" Ich mach nichts! "
   Menu$(2,1)=" Ja, drück mich! "
   Menu ON
End Proc
Procedure ALTER
   Shared FLAG
   Menu Del
   If FLAG=0 Then Menu Bar 1 : FLAG=1 Else Menu Tline 1 : FLAG=0
   SET_MEN
Endproc
```

MENU INACTIVE *(Schaltet Menüpunkt ab)*

MENU INACTIVE Ebene
MENU INACTIVE(,,)

Wie der Name schon sagt, setzt MENU INACTIVE eine Reihe von Optionen in Ihrem Menü außer Kraft. Alle Versuche, diese Punkte danach zu wählen, werden völlig ignoriert. Ebene ermöglicht es Ihnen, einen ganzen Abschnitt des Menüs stillzulegen. Durch Eingabe der Parameter (,,) können Sie aber auch einzelne Menüoptionen abschalten. Diese Parameter geben die genaue Position an, an der Ihr Menüpunkt in der Menüstruktur steht.

Beachten Sie hier bitte, daß die Menüoptionen, die Sie mit dieser Anweisung abschalten, sofort durch die Zeichenkette INACTIVE$ ersetzt werden, die Sie in Ihrer ursprünglichen Menüdefinition angegeben haben. Wenn hier keine Eingabe gemacht wurde, werden die Menüoptionen, die nicht verfügbar sind, kursiv dargestellt.

MENU ACTIVE *(Aktiviert einen Menüpunkt)*

MENU ACTIVE Ebene
MENU ACTIVE(,,)

MENU ACTIVE kehrt einfach die Wirkung eines vorher eingebenen MENU INACTIVE Befehls um. Durch den Parameter *Ebene* wählen Sie eine ganze Menüebene aus, die neu gestartet werden soll. (,,) aktiviert einen einzigen Punkt Ihres Menüs.

Die gewählten Optionen werden automatisch wieder unter Verwendung Ihrer ursprünglichen Titelketten angezeigt, nachdem Sie diese Anweisung aufgerufen haben.

Bewegliche Menüs

AMOS Menüs können an jeder beliebigen Stelle auf dem Bildschirm des Amiga angezeigt werden. Die Menüs können entweder ausdrücklich über das Programm oder direkt durch den Anwender bewegt werden.

MENU MOVABLE *(Aktiviert die automatische Menübewegung)*

MENU MOVABLE Ebene
MENU MOVABLE(,,)

MENU MOVABLE teilt dem Menüsystem mit, daß die Menüpunkte auf der vorgegebenen Ebene durch den Anwender bewegt werden können - das ist der Default.

Die zweite Form dieser Anweisung erlaubt es Ihnen, den Status jedes einzelnen Menüpunktes festzulegen. Die Parameter in Klammern können dabei jede beliebige Position in der Menüstruktur angeben.

Jedes Menü kann dadurch neu positioniert werden, daß man den Mauszeiger auf den **ersten** Punkt im Menü stellt und die linke Maustaste drückt. Dann erscheint ein rechteckiger Kasten um den jeweiligen Menüpunkt und man kann ihn zu jeder beliebigen Stelle auf dem aktuellen Schirm schieben. Wenn Sie die linke Taste loslassen, wird das Menü mit allen zugehörigen Optionen an der neuen Position erstellt.

Beachten Sie bitte, daß Sie mit diesem Befehl die Anordnung von Punkten, die unter dieser Ebene liegen, nicht ändern können. Wenn Sie die einzelnen Menüoptionen manipulieren möchten, dann müssen Sie dazu den besonderen MENU ITEM Befehl verwenden. Wie dieses System funktioniert, können Sie sich in **BEISPIEL 16.12** näher ansehen.

MENU STATIC *(Bringt ein Menü an einer Stelle fest an)*

MENU STATIC Ebene
MENU STATIC(,,)

MENU STATIC definiert, daß der Anwender das Menü auf der angegebenen Ebene nicht verschieben kann.

Ein Problem der beweglichen Menüs ist, daß sich der Speicherplatz, den sie während eines Programmablaufs benötigen, verändert. Wenn Ihre Menüs also besonders umfangreich sind, oder der Speicher sowieso etwas knapp wird, dann kann ein unbedachter Schritt des Anwenders zur Fehlermeldung *Kein Speicherplatz* mehr führen, und so Ihr Programm zu einem vorzeitigen Ende bringen. Mit Hilfe der Anweisung MENU STATIC können Sie dieses Problem völlig vermeiden.

MENU ITEM MOVABLE *(Bewegt einzelne Menüoptionen)*

MENU ITEM MOVABLE Ebene
MENU ITEM MOVABLE(,,)

Dieser Befehl ähnelt dem oben erklärten MENU MOVABLE im Prinzip. Der Unterschied besteht jedoch darin, daß Sie hier die Anordnung der verschiedenen Optionen auf einer bestimmten Ebene verändern können. Also können alle Punkte in einem bestimmten Menübalken vom Anwender einzeln neu positioniert werden.

Normalerweise ist es nicht erlaubt, die Punkte außerhalb des aktuellen Menübalkens zu verschieben, aber durch den Befehl MENU SEPARATE kann man diese Regel aufheben.

Damit jedoch die einzelnen Menüpunkte beweglich sind, muß man auch den **ganzen** Menübalken bewegen können. Wenn Sie also dem Menü durch MENU STATIC einen festen Platz zugewiesen haben, ist diese Anweisung völlig wirkungslos. Darüber hinaus können Sie den ersten Punkt in einem Menübalken nicht allein bewegen, denn das verschiebt die ganze Zeile. Eine weitere Nebenwirkung besteht darin, daß das Verschieben des **letzten** Menüpunktes den Umfang Ihres Menübalkens bleibend verringert. Es gibt die folgenden beiden Lösungsmöglichkeiten:

- Umrahmen Sie Ihren gesamten Balken folgendermaßen mit einem Rechteck:

  Menu$(1,1)=,,,"(Bar 40,100)(LOc 0,0)"

  Dabei stellt MENU$(1,1) den ersten Punkt in Ihrem aktuellen Menübalken dar.

- Fixieren Sie den letzten Punkt mit MENU ITEM STATIC.

MENU ITEM STATIC *(Fixiert einen Menüpunkt)*

MENU ITEM STATIC Ebene
MENU ITEM STATIC(,,)

Durch diesen Befehl werden ein oder mehrere Menüpunkte fest an einer bestimmten Stelle angebracht. Dies ist auch der Default.

MENU SEPARATE *(Spaltet eine Reihe von Menüpunkten)*

MENU SEPARATE Ebene
MENU SEPARATE(,,)

MENU SEPARATE weist AMOS an, alle Punkte auf der aktuellen Menüebene voneinander zu trennen. So wird nun jeder Punkt ganz unabhängig für sich behandelt. Wenn Sie keine Hintergrundkette (background string) definiert haben, dann beträgt der Offset zu dem jeweils darüberliegenden Menüpunkt immer genau zwei Pixel. Dadurch entsteht ein interessanter Staffeleffekt, den Sie aber durch Bearbeiten des Menüs mit dem MENU-Zusatzprogramm auch entfernen können.

Die Parameter, die Sie wahlweise bei dieser Anweisung eingeben können, ermöglichen es Ihnen, einen Menübalken an jeder beliebigen Stelle in der Zeile zu spalten. Wenn Sie einen Punkt einmal abgetrennt haben, dann finden die Befehle MENU MOVABLE anstatt der ITEM-Anweisungen Anwendung.

MENU LINKED *(Verbinden von einer Reihe von Menüs)*

MENU LINKED level
MENU LINKED(,,)

Durch diese Anweisung werden zwei oder mehr Menüpunkte miteinander verbunden. Sie stellt das Gegenteil des Befehls MENU SEPARATE dar.

= X MENU *(Gibt die X-Grafikkoordinate eines Menüpunktes an)*

x=x MENU(,,)

Durch die Funktion X MENU können Sie die Position eines Menüpunktes in Bezug zur vorherigen Option auf dem Bildschirm abfragen. Anhand dieser Information können Sie dann so leistungsstarke Menüs wie das in **BEISPIEL 16.13** implementieren.

=Y MENU *(Gibt die Y-Grafikkoordinate eines Menüpunktes an)*

x=y MENU(,,)

Weist die Y-Koordinate einer Menüoption aus. Beachten Sie, daß alle Koordinaten in Bezug zum vorherigen Menüpunkt erfaßt werden. Es handelt sich also nicht um normale Bildschirmkoordinaten.

Das Bewegen eines Menüs in einem Programm

MENU BASE *(Bewegt den Anfangspunkt eines Menüs)*

MENU BASE x,y

Dieser Befehl versetzt den Anfangspunkt der ersten Ebene Ihres Menüs auf die absoluten Bildschirmkoordinaten *x,y*. Alle untergeordneten Menüpunkte werden an ihrer aktuellen Position im Verhältnis zum oberen Abschnitt des Menüs angezeigt. In **BEISPIEL 16.14** können Sie den Befehl MENU BASE in Aktion sehen.

SET MENU *(Bewegt ein Menü)*

SET MENU (,,) TO x,y

Mit SET MENU können Sie die Koordinaten der linken oberen Ecke eines Menüpunktes bestimmen. Diese Koordinaten werden in Bezug zur vorherigen Ebene erfaßt. Der Anfangspunkt für das gesamte Menü (Koordinaten 0,0) kann durch den Befehl MENU BASE festgelegt werden.

Alle Ebenen des Menüs, die unter Ihrem Menü liegen, werden von dieser Anweisung ebenfalls verschoben. Ihre jeweiligen Positionen bleiben dabei unverändert. Da *x,y* auch negativ sein können, ist es möglich, die einzelnen Punkte in einem Menübalken in Form einer Schalttafel anzuordnen - siehe dazu auch **BEISPIEL 16.15**.

Das Anzeigen eines Menüs an der Cursor-Position

MENU MOUSE *(Zeigt das Menü unter der Maus an)*

MENU MOUSE ON/OFF

Die MENU MOUSE Funktionen zeigen automatisch alle Menüs an, angefangen bei der aktuellen Cursor-Position. Die Mauskoordinaten werden zu den Koordinaten von MENU BASE addiert, damit die Endposition errechnet werden kann. Auf diese Weise kann das Menü auch - falls erforderlich - in einem bestimmten Abstand vom Mauszeiger plaziert werden. Siehe **BEISPIEL 16.16**.



17: Der Ton macht die Musik

Mit dem Sound-System des Amiga kann man Sound-Effekte in Stereo hervorrufen, die vor ein paar Jahren noch völlig unvorstellbar schienen. Was da allein aus dem Lautsprecher Ihres Fernsehers ertönt, ist schon beeindruckend, aber wenn Sie Ihren Amiga erst an eine Hi-Fi-Anlage anschließen, dann wackeln die Wände!

Nachdem Sie AMOS jetzt schon etwas kennengelernt haben, können Sie sich natürlich gut vorstellen, daß wir hier weit über den einfachen BEEP-Befehl hinausgehen. Wir haben nämlich so ziemlich alles eingebaut, was Sie brauchen, um in Ihren Spielen ganz umwerfende Sound-Effekte zu erzeugen. Alle AMOS Sound- Befehle werden völlig unabhängig von Ihren Basic-Programmen ausgeführt. Auf diese Art können Ihre Sound-Tracks ununterbrochen abgespielt werden, ohne irgendeine Auswirkung auf die Qualität des Spiels selbst zu haben.

Sie können mit jeder der verfügbaren Sampling-Kassetten Samples (Instrumente) erzeugen, die ganz einfach durch die Anweisung SAMPLAY wieder abgespielt werden können. Jedes Sample kann in vielen verschiedenen Geschwindigkeiten ausgegeben werden und wiederholt als Programmschleife eingesetzt werden. Es ist sogar möglich, ein Sample als Note zu spielen.

Musik kann auch von einem anderen Software-Paket wie SONIX, SOUNDTRACKER oder GMC konvertiert werden. Das AMOS Musiksystem ist intelligent und stoppt sofort automatisch, wenn ein Ton über den aktuellen Kanal geschickt wird. Auf diese Art können Sie problemlos Samples und Musik im selben Tonkanal verbinden, ohne dabei unerwünschte Störeffekte zu riskieren.

Jedes Lied kann bis zu 256 einzelne Instrumente enthalten, und die einzige Begrenzung der Anzahl der Lieder ist durch den verfügbaren Speicher gegeben. Um den Speicherverbrauch auf ein absolutes Minimum zu beschränken, werden alle Melodien aus einer Reihe von verschiedenen Mustern zusammengesetzt. Nachdem ein solches Muster einmal erzeugt wurde, ist es von jeder beliebigen Stelle Ihrer Musik aus zugänglich. Und das verbraucht dann jeweils nur ein paar Bytes. Daher reicht es aus, wenn Sie nur eine kleine Anzahl an Schlüsselmustern definieren. Sie können mit ihnen dann Dutzende von verschiedenen Melodien erzeugen, ohne dabei Speicher zu fressen.

Das beste am AMOS Musiksystem ist jedoch, daß es leicht erweitert werden kann. Sie erhalten den gesamten Quelltext auf der Datendiskette und können ihn ganz nach Wunsch untersuchen oder ändern. Und so verpassen Sie dann bei zukünftigen Entwicklungen in der Amiga Musikszene nicht so leicht den Anschluß.

Einfache Sound-Effekte

Wir wollen erst einmal mit einer Übersicht der in AMOS-Basic eingebauten Sound-Effekte beginnen. Sie entsprechen dem BEEP-Befehl in Amiga Basic.

BOOM *(Erzeugt ein explosionsartiges Geräusch)*

BOOM

Peng! Schon bist Du tot! Mit BOOM können Sie Ihre Spiele durch den entsprechenden Sound-Effekt in Stereo begleiten.

Es war eigentlich bisher immer recht schwierig, diese Art von unvermitteltem Geräusch auf dem Amiga zu erzeugen. AMOS erzielt jedoch mittels eines raffinierten Interrupt-Systems einen äußerst realistischen Explosionseffekt. Überzeugen Sie sich doch anhand eines Beispiels davon:

```
Boom : Print "Du bist TOT!"
```

SHOOT *(Erzeugt ein Geräusch, das wie ein Schuß klingt)*

SHOOT

Der Befehl SHOOT ruft einen Effekt hervor, der wie ein Schuß klingt. Wie BOOM hält auch SHOOT Ihr Programm überhaupt nicht an. Wenn Sie also mehrere Schüsse hintereinander abfeuern, ist es vielleicht ratsam, mit dem Befehl WAIT eine kleine Verzögerung einzubauen.

```
Shoot : Wait 6 : Shoot : Print "Du bist TOT!"
```

BELL *(Einfacher Glockenton)*

BELL [f]

BELL erzeugt einen reinen Ton mit der Frequenz *f*. *f* bestimmt die Tonlage, von 1 (sehr tief) bis 96 (sehr hoch). Hier ist ein Beispiel dazu:

```
Bell : Wait 40 : Rem Warten bis Ton verklingt
For F=1 To 96
   Bell F:Wait F/10+1 : Rem Verändert Verzögerung zusammen mit Frequenz
Next F
```

Die Tonkanäle

Die Amiga-Hardware kann problemlos bis zu vier verschiedene Töne gleichzeitig erzeugen. Auf diese Art können Sie Ihre Sound-Effekte durch angenehme Obertöne begleiten.

Jeder Ton kann durch eine von vier Stimmen ausgegeben werden. Sie sind von 0 bis 3 durchnumeriert. Sie können sich diese Stimmen als unabhängige Instrumente

vorstellen, die ihre eigenen Tonfolgen, Samples oder Musikstücke spielen können. Alle vier Stimmen werden intern miteinander kombiniert und erzeugen dann so den Klang, der schließlich aus Ihrem Lautsprecher kommt.
Mit den AMOS Sound-Anweisungen können Sie Ihren Sound mit jedem beliebigen Arrangement der Stimmen abspielen lassen. Alle AMOS Sound-Befehle beruhen auf der üblichen Eingabemethode von Stimmwerten. Jeder Stimme wird dabei ein bestimmtes Bit in einem VOICE-Parameter folgendermaßen zugeordnet:

Bit 0-> Stimme 0
Bit 1-> Stimme 1
Bit 2-> Stimme 2
Bit 3-> Stimme 3

Die gewünschten Stimmen können Sie dadurch aktivieren, daß Sie das entsprechende Bit auf 1 stellen. Hier ist ein Reihe der häufigsten Werte um die Sache etwas zu vereinfachen.

Wert	Eingesetzte Stimmen	Wirkung
15	0, 1, 2, 3	Verwendet alle vier Stimmen.
9	0, 3	Diese Stimmen werden miteinander kombiniert und durch den linken Lautsprecher ausgegeben.
8	3	
6	2, 4	Ausgabe erfolgt durch den rechten Lautsprecher.
4	2	
2	1	
1	0	

Um den so erzeugten Sound-Effekten wirklich gerecht zu werden, müssen Sie wahrscheinlich wirklich Ihren Amiga an ein Hi-Fi-System anschließen. Die meisten Fernseher können nämlich die Bandbreite der Sound-Effekte, die man mit der erstaunlichen Amiga-Hardware erzeugen kann, nicht wiedergeben.

VOLUME *(Verändert die Lautstärke)*

VOLUME [v,] Intensität

VOLUME verändert die Lautstärke der Töne, die über einen oder mehrere der Tonkanäle ausgegeben werden.

Intensität bezieht sich darauf, wie laut dieser Ton nun sein soll. Normalerweise erstreckt der Bereich von 0 (stumm) bis 63 (sehr laut). Der Default setzt die Intensität aller vier verfügbaren Stimmen gleich. Wenn Sie hier nun einen neuen Wert für die Lautstärke eingeben, so gilt dieser für alle zukünftigen Sound-Effekte und auch für die Musik.

Über den Parameter *v* können Sie die Lautstärke für jede Stimme einzeln einstellen. *v gibt* also an, welche Kombination von Stimmen durch diese Anweisung reguliert werden sollen. Die zweite Option wird nur für die Sound-Effekte eingesetzt. Sie hat auf die Musik, die Sie spielen, überhaupt keine Auswirkung. Die Stimmen werden über eine Bitmap im Standardformat ausgewählt, dabei stellt jedes Bit den Status eines bestimmten Tonkanals dar. Wenn das Bit auf 1 gestellt wird, dann wird die Lautstärke dieser Stimme verändert, sonst bleibt sie völlig unberührt. Hier sind einige Beispiele dazu:

```
Volume %0001,63 : Boom : Wait 100 : Rem Stellt Kanal 1 auf Lautstärke 63
Volume %1110,10 : Boom : Wait 50 : Rem Kanäle 2,3,4 haben Lautstärke 10
Play 40,0 : Wait 30
Volume 50 : Play 40,0
```

Gesampelte Sounds

Wenn Sie alle Sound-Effekte, die Sie brauchen, direkt in Ihrem Computer erzeugen müßten, dann würden Sie versuchen, das Unmögliche möglich zu machen. In der Praxis ist es oft leichter, echten Sound aus einer externen Quelle - wie zum Beispiel einem Kassettenrekorder - zu nehmen und ihn in eine Reihe von Werten umzusetzen, die dann in den Speicher Ihres Computers geladen werden können.

Diese Werte stellen die Lautstärke eines bestimmten Sound-Samples dar. Wenn diese Werte dann schnell über die Sound-Chips des Amiga zurückgespielt werden, erhalten Sie eine realistische Wiedergabe des ursprünglichen Sounds. Diese Technik stellt die Grundlage für alle gesampelten Sound-Effekte in den meisten aktuellen Computerspielen dar.

Wenn Sie Ihre eigenen Samples erzeugen möchten, müssen Sie sich eine besondere Hardware-Ausrüstung beschaffen, die man SAMPLER nennt. Die Arbeit mit diesen Sampler-Systemen macht zwar viel Spaß, aber unbedingt erforderlich sind sie nicht. AMOS-Basic kann nämlich jedes vorhandene Sound-Sample sehr gut abspielen, und braucht dazu keine teuren Add-Ons.

Es sind zur Zeit Hunderte von Public-Domain-Samples auf dem Markt, die Ihnen die meisten der Sound-Effekte bieten, die Sie für Ihre Spiele brauchen können. Auf der AMOS Datendiskette haben wir sogar eine Auswahl von nützlichen Samples untergebracht, damit Sie mit ihnen ein bißchen experimentieren können.

SAM PLAY *(Spielt ein Sound-Sample aus der AMOS Sample-Bank)*

SAM PLAY s
SAM PLAY v,s
SAM PLAY v,s,f

Die Anweisung SAM PLAY spielt einen gesampelten Sound direkt über Ihr Lautsprechersystem ab. Alle Samples werden normalerweise in der Speicherbank mit der Nummer 5 gespeichert, aber das können Sie mit dem Befehl SAM BANK beliebig ändern.

s stellt die Nummer des Samples dar, das gespielt werden soll. Die Anzahl der Samples, die Sie in einer Bank speichern können, ist eigentlich nur durch den verfügbaren Speicherplatz begrenzt. Wenn Sie diese Anweisung auf Ihre eigenen Samples anwenden möchten, dann müssen Sie sie erst in eine AMOS Speicherbank stellen. Nähere Einzelheiten dazu finden Sie am Ende dieses Kapitels.

v ist eine Bitmap, die eine Reihe von Stimmen enthält, auf die Ihr Sample zugreift. Wie üblich gibt es auch hier für jede der Stimmen ein bestimmtes Bit. Wenn Sie Ihre Samples über die entsprechende Stimme abspielen möchten, dann stellen Sie einfach nur das betreffende Bit auf 1. Mehr Information darüber finden Sie in der Erklärung der TONKANÄLE weiter vorne in diesem Kapitel.

f gibt die Abspielgeschwindigkeit für Ihr Sample, gemessen in *Hertz*, an. Dies bestimmt die Anzahl der Samples, die pro Sekunde gespielt werden sollen. Die typischen Geschwindigkeiten der Samples liegen zwischen 4000 - für Geräusche wie Explosionen zum Beispiel - und 10000, für verständliche Sprachsimulationen. Durch Verändern der Abspielgeschwindigkeit stehen Ihnen eine Vielzahl von Möglichkeiten zur Verfügung, die Höhe Ihres Tons beliebig anzupassen. Sie können also mit einem Sample eine große Reihe von verschiedenen Effekten erzeugen. Dazu folgende Beispiele:

```
Rem Lädt Sample-Bank mit einigen Samples von der AMOS Datendiskette
Load "AMOS_DATA:SAMPLES/SAMPLES.ABK"
For S=1 to 11
   Locate 0,0:? "Spielt ein Sample ";S
   Sam Play S
Locate 0,24 : Centre "<Zum Weitermachen Taste drücken>" : Wait Key : Cline
Next S
Wait Key
Sam Play 1,11 : Wait 5 : Sam Play 2,11 : Rem Einfache Echo-Wirkung
Wait Key
Sam Play 1,1,2000 : Rem Niedrige Tonlage
Wait Key
Sam Play 1,1,15000 : Rem Hohe Tonlage
```

Eine weitere Demonstration dieses Befehls finden Sie in **BEISPIEL 17.1** im MANUAL-Unterverzeichnis.

SAM BANK *(Verändert die aktuelle Sample-Bank)*

SAM BANK n

SAM BANK bestimmt eine neue Speicherbank, die nun für Ihre Arbeit mit den Samples eingesetzt wird. Alle zukünftigen SAM PLAY Anweisungen entnehmen jetzt die Sounds direkt dieser Bank.

Diese Funktion können Sie dazu verwenden, mehrere Sample-Sätze gleichzeitig zu speichern. Sie können dann jederzeit zwischen diesen Samples hin- und herspringen, indem Sie einfach schnell den Befehl SAM BANK aufrufen.

SAM RAW *(Spielt ein Sample aus dem Speicher)*

SAM RAW Stimme,Adresse,Länge,Frequenz

SAM RAW spielt ein Originalsample ab, das irgendwo im Amiga Speicher gespeichert ist. *Stimme* ist ein Bitmuster im Standardformat, das diejenigen Stimmen, die Ihr Sample einsetzen soll, bestimmt. Jedes Bit in dem Muster wählt dabei einen Kanal aus. Weitere Einzelheiten dazu finden Sie im Abschnitt über die Tonkanäle.

Adresse enthält die Adresse Ihres Samples. Normalerweise bezieht sich diese Angabe auf eine Position in der vorhandenen AMOS Speicherbank. Länge gibt die Länge des Samples an, das Sie abspielen möchten. Frequenz zeigt entweder in Samples pro Sekunde oder Hz an, mit welcher Geschwindigkeit das Sample abgespielt werden soll. Die Abspielgeschwindigkeit kann sich dabei von der Geschwindigkeit, mit der das Sample ursprünglich aufgenommen wurde, durchaus unterscheiden.

Mit SAM RAW können Sie die normalen Amiga-Samples direkt über Ihren Lautsprecher abspielen, ohne eine besondere Speicherbank erzeugen zu müssen (siehe *Das Erzeugen einer Sample-Bank*). Sie müssen jetzt Ihre Samples im Speicher ganz selbständig verwalten und die Sample-Parameter mit der Hand eingeben. SAM RAW eignet sich wunderbar zum Durchstöbern Ihrer Diskettensammlung. Laden Sie zum Beispiel eine Datei mit BLOAD in eine Bank und spielen Sie die Daten dann mit SAM RAW ab. Wenn Sie ein bißchen Glück haben, dann kommen dabei ganz interessante Töne heraus. Hier sind ein paar Beispiele:

```
Reserve as work 10,55000
Bload "Samples/Samples.abk",start(10)
Sam Raw 15,start(10),length(10),10000
```

SAM SWAP *(Definiert das automatische Wechseln von Samples)*

SAM SWAP Stimmen TO Adresse,Länge

Diese Anweisung intialisiert das automatische Wechseln von Samplen. Der Wechsel findet allerdings erst statt, wenn der Sample im Speicher vollkommen abgespielt wurde.

Stimmen ist ein Bit-Muster, welches die betroffenen Stimmen definiert, genauso wie in der SAM RAW Anweisung.

Adresse ist die Adresse des nächsten Samples zum Abspielen. Sie muß einen Speicherbereich im Chip-Speicher angeben.

Länge gibt die Länge des nachfolgenden Samples an.

=SAM SWAPPED *(Überprüft ob Sample-Wechsel stattgefunden hat.)*

=SAM SWAPPED(Stimme)
Die Funktion SAM SWAPPED überprüft ob ein zuvor mit SAM SWAP definierter Wechsel der Samples stattgefunden hat. Sie ist deshalb gerade bei mehrstimmigen Samples unabdingbar.

Stimme ist die Nummer der Stimme, über die Sie Informationen bekommen wollen. Verwechseln Sie dies nicht: *Stimme* ist **kein** Bitmuster, sondern eine Zahl von 0-3.

Es wird TRUE (=WAHR), also -1 zurückgegeben, wenn der automatische Wechsel schon erfolgt ist, also der mit SAM SWAP definierte Speicherbereich in diesem Moment abgespielt wird. Ansonsten gibt Sie 0, also FALSE (=Falsch) zurück.

SAM STOP *(Beendet das Abspielen eines Samples)*

SAM STOP (Stimmen)

Beendet das Abspielen eines Samples. *Stimmen* gibt ein Bitmuster (siehe SAM RAW) an, welches die zu desaktivierenden Stimmen angibt.

SAM LOOP *(Wiederholt ein Sample)*

SAM LOOP ON/OFF

Die Anweisung SAM LOOP teilt AMOS-Basic mit, daß alle folgenden Samples ständig wiederholt werden sollen. Hier sind einige Beispiele:

```
Rem Lädt die Sample-Bank mit einigen Samples von der AMOS Datendiskette
Load "AMOS_DATA:SAMPLES/SAMPLES.ABK"
Sam Loop On
For S=1 to 11
   Locate 0,0 : Print "Spielt ein Sample ";S
   Sam Play S
   Locate 0,24 : Centre"<Zum Weitermachen eine Taste drücken>" : Wait Key : Cline
Next S
Sam Loop Off
```

Diese Schleifenbildung können Sie ganz einfach jederzeit durch Aufrufen von SAM LOOP OFF wieder abschalten.

SLOAD *(Lädt einen Teil eines Samples in den Speicher)*

SLOAD Filenummer, Länge TO Adresse

Die SLOAD Anweisung ist eine ausgebaute Version des BLOAD Befehls (siehe Kapitel 22). Sie ermöglicht es, extrem lange Samples teilweise in den Speicher zu laden. Dies ermöglicht Ihren Multimedia-Anwendungen ganz neue Dimensionen. SLOAD ist bei einem Diskettenlaufwerk nicht sinnvoll einzusetzen. Die Übertragungsgeschwindigkeit erlaubt nur niederfrequenzige Samples. Dagegen können Benutzer mit Festplatte oder CD-ROM gigantische Datenmengen einladen.

Filenummer ist die Identifikationsnummer eines zuvor geöffneten Samplefiles. Benutzen Sie die OPEN IN Filenummer,"Filename" Funktion.

Länge gibt die Länge des zu ladenden Bereiches an. Ist er größer als die tatsächliche Größe der Datei, so wird diese (ohne Fehlermeldung), bis zu deren Ende eingeladen. Eine Fehlermeldung erhalten Sie, wenn Sie daraufhin nochmals von der gleichen Datei einladen wollen, obwohl deren Ende erreicht ist.

Adresse gibt den Zielbereich im Speicher an. Dieser wird **NICHT** von AMOS reserviert; Sie müssen also selbst Sorge dafür tragen, wo Ihre Samples landen.

Der Vorteil dieses Befehls liegt darin, daß die Sample-Datei wie jede andere sequentielle Datei behandelt wird. Das heißt, daß Sie mit dem POF() Befehl die Leseposition innerhalb der Datei beliebig verschieben können. Der SLOAD Befehl beschränkt sich also nicht nur auf Samples, auch andere Daten können so einfach eingeladen werden.

Das Erzeugen einer Sample-Bank

Wenn Sie mit SAM PLAY Ihre eigenen Samples abspielen möchten, müssen Sie sie zuerst in eine Speicherbank laden. Das können Sie mit Hilfe des Programmes SAMPLE BANK MAKER erreichen, das Sie auf der AMOS Datendiskette finden.

Nach dem Programmstart zeigt Ihnen SAMMAKER einen normalen AMOS Datei-Selektor. Geben Sie hier nun den Namen des ersten Samples ein, das in Ihre neue Bank geladen werden soll und drücken Sie ENTER. Kann AMOS nun keine Sample-Geschwindigkeit finden, müssen Sie sie mit der Hand eingeben. Wenn Sie hier einen Fehler machen, spielt das im Grunde keine Rolle, denn Sie können ja Ihre Samples problemlos mit jeder beliebigen Geschwindigkeit wieder abspielen. Nach eine kleinen Verzögerung fragt AMOS Sie dann nach dem nächsten Sample, das Sie in dieser Bank unterbringen möchten. Wenn Sie am Ende Ihrer Samples angekommen sind, so tippen Sie SAVE in den Datei-Selektor ein, und Ihre Samples werden auf der Diskette gespeichert. Sie werden dabei automatisch nach dem Dateinamen des Ziels für Ihre neue Bank gefragt. Das können Sie jetzt mit dem LOAD-Befehl folgendermaßen in AMOS-Basic eingeben:

```
Load "samples.abk"
Load "samples.abk",6 : Rem lädt Samples in Bank 6
```

Musik

Mit dem AMOS Musiksystem können Sie Ihre Spiele ganz leicht mit einer interessanten Hintergrundspur versehen. Die Musik kann aus einer Vielzahl von Quellen stammen, einschließlich GMC, SOUNDTRACKER oder SONIX.

Um diese Musik nun in das spezielle AMOS-Format zu konvertieren, müssen Sie

eines der Übersetzungsprogramme auf der AMOS Datendiskette einsetzen. GMC- Musik sollte zum Beispiel über das Icon SAVE DATA gespeichert worden sein, da auf diese Art sowohl die Musik wie auch die Definitionen der Instrumente in einer einzigen, großen Datendatei gespeichert werden.

MUSIC *(Spielt ein Musikstück)*

MUSIC n

Der AMOS MUSIC-Befehl startet ein Musikstück aus der Musikbank (das ist die Bank Nummer 3). Diese Musik wird unabhängig von Ihrem Basic-Programm abgespielt, und hat auf das Programm überhaupt keine Auswirkungen.

Normalerweise ist es möglich, mehrere vollständige Arrangements in derselben Bank zu speichern. Jeder Komposition wird dabei ihre eigene Musiknummer zugewiesen. Die einzige Ausnahme zu dieser Regel ist die mit GMC erzeugte Musik, denn hier können Sie nur jeweils immer ein Lied in die Bank stellen. Dazu folgendes Beispiel:

```
Rem Lädt ein Musikstück von der AMOS Datendiskette
Load "MUSIC/musicdemo.abk"
Music 1
```

Das AMOS Musiksystem ist intelligent und unterbricht Ihre Musik automatisch für alle folgenden Sound-Effekte aus dem aktuellen Tonkanal. Wenn das betreffende Geräusch zu Ende ist, dann beginnt Ihre Melodie wieder dort, wo sie ausgesetzt hat.

Es können bis zu drei verschiedene Melodien gleichzeitig gestartet werden. Jeder neue Musikbefehl hält das gerade gespielte Lied an, und schiebt seinen Status in einen Stapel. Wenn das neue Lied dann fertigabgespielt wurde, beginnt die vorhergehende Musik wieder dort, wo sie unterbrochen wurde.

MUSIC STOP *(Stoppt einen bestimmten Musikabschnitt)*

MUSIC STOP

MUSIC STOP hält das aktuelle Musikstück an. Wenn ein anderes Musikstück aktiv ist, so wird es sofort wieder neu gestartet.

MUSIC OFF *(Schaltet die gesamte Musik ab)*

MUSIC OFF

Mit dem Befehl MUSIC OFF wird Ihre Musik völlig abgeschaltet. Um die Musik danach wieder aufzunehmen, müssen Sie Ihre ursprüngliche Folge von MUSIC- Anweisungen noch einmal von vorn ausführen.

TEMPO *(Verändert das Tempo eines Musik-Samples)*

TEMPO s

TEMPO verändert die Geschwindigkeit der Melodie, die Sie gerade mit dem MUSIC-Befehl abspielen. s stellt dabei das neue Tempo dar und kann zwischen 1 (sehr langsam) und 100 (sehr schnell) liegen. Nicht alle Instrumente können jedoch mit dieser Höchstgeschwindigkeit gespielt werden. In der Praxis liegt die Grenze in den meisten Fällen eher bei einem Wert von 50.

Sehen Sie sich das doch an einem Beispiel an. Legen Sie die AMOS Datendiskette in das aktuelle Laufwerk ein und tippen Sie:

```
Load "AMOS_DATA:MUSIC/musicdemo.abk" : Rem Lädt Musik
Music 1 : Rem Spielt Musik 1
Tempo 35 : Rem Stelle Musik auf sehr schnell
Tempo 5 : Rem Musik fängt jetzt sehr langsam zu Spielen an
```

Beachten Sie dabei, daß mit GMC erzeugte Musik oft Sprungmarken aufweist, die das Tempo direkt in dem Arrangement bestimmen. Diese Sprungmarken haben vor den in AMOS-Basic gesetzten Tempoangaben Vorrang. Sie sollten sie also nicht in Ihrer eigenen Musik einsetzen.

MVOLUME *(Bestimmt die Lautstärke eines Musikstücks)*

MVOLUME n

Diese Anweisung verändert die Lautstärke des gesamten Musikstückes zur Intensität *n*. *n* kann dabei zwischen 0 (stumm) und 63 (sehr laut) liegen.

VOICE *(Aktiviert eine oder mehrere Stimmen eines Musikstückes)*

VOICE Maske

VOICE aktiviert eine oder mehrere Stimmen des Musikstückes unabhängig voneinander. Normalerweise weist jede Stimme ihre eigene Melodie auf. In Ihrem Lautsprecher werden diese einzelnen Melodien dann zu einem Ganzen verbunden.

Maske ist eine Bitmaske im normalen AMOS-Format, die angibt, welche Stimmen Sie spielen möchten. Jedes Bit stellt den Status einer Stimme des Musikstücks dar. Wenn dieses Bit auf 1 gestellt wird, so wird die entsprechende Stimme gespielt, ansonsten wird sie nicht eingesetzt. Hier sind einige Beispiele dazu:

```
Load "MUSIC/musicdemo.abk" : Rem Lädt Musik
Music 1 : Rem Spielt Musik 1
```

```
For V=0 To 15
   Locate 0,0 : Print "Stimme ";V
   Voice V
   Wait 100
Next V
Direct
Voice %0001 : Rem Aktiviert Stimme 0
Voice %0010 : Rem Stimme 1
Voice %1001 : Rem Stimmen 3 und 0
Voice %1111 : Rem Stimme 4
```

=VUMETER *(Messen der Lautstärke)*

s=VUMETER(v)

Die VUMETER-Funktion fragt die Stimme *v* ab und weist die Lautstärke der Note, die von Ihrem Musikstück gerade gespielt wird, aus. *s* ist ein Wert, der die Intensität angibt, und zwischen 0 und 63 liegt. *v* ist die Nummer der Stimme, die abgefragt werden soll (von 0 - 3).

Mit dieser Funktion können Sie sogar Ihre Sprites zu einem Musikstück tanzen lassen! Laden Sie **BEISPIEL 17.2** und sehen Sie sich eine Demonstration dazu an.

Es gibt außerdem eine AMAL-Version dieser Anweisung, durch die Sie über Interrupts Echtzeit-VUmeter erzeugen können. Im Abschnitt zu dem VU-Befehl finden Sie weitere Einzelheiten darüber.

Direktes Abspielen von Musiken

Die normalen AMOS Musiken müssen vor dem Abspielen durch ein Konvertierungsprogramm in das AMOS Musik Bank Format gebracht werden. Dies ist für einige Anwendungen etwas umständlich. Deshalb bietet AMOS auch die Möglichkeit, jeden StarTracker Sound direkt abzuspielen.

TRACK LOAD *(Lädt ein Modul)*

TRACK LOAD "Modul",bank_nummer

Lädt ein Soundtracker/Noisetracker/Startracker Modul in eine Speicherbank.

Bank_nummer gibt die Nummer der AMOS Bank an, in die das Modul geladen werden soll. Die entsprechende Bank wird zunächst gelöscht, dann neu reserviert. Diese Bank liegt immer im CHIP-Speicher, bei großen Musikstücken kann es also zu Speicherproblemen kommen.

Nachdem die Bank geladen wurde, wird *bank_nummer* als Default-Wert für TRACK PLAY gesetzt.

Sollte AMOS ein synthetisches Instrument von StarTracker finden (welches die Endung .NT haben muß!), so wird es ebenfalls in diese Bank geladen. Darüber brauchen Sie sich keine Sorgen machen.

TRACK PLAY *(Spielt ein Modul ab)*

TRACK PLAY [Bank[,Pattern]]

Spielt eine zuvor geladene Modulbank ab. Wird der Parameter Bank weggelassen, bezieht sich die Anweisung auf die zuletzt geladene Bank. Wurde während des Programmablaufes keine Bank geladen, so wird die Bank 6 abgespielt.

Pattern gibt das Startpattern der Musik an. **Vorsicht** : AMOS überprüft diese Angabe nicht. Wird ein zu hoher Wert eingesetzt, kann das Programm abstürzen. Wird *Pattern* weggelassen, wird die Bank von Anfang an abgespielt.

TRACK STOP *(Hält die aktuelle Musik an)*

TRACK STOP

Die aktuelle Musik wird sofort unterbrochen und beendet.

TRACK LOOP *(Endloses Abspielen)*

TRACK LOOP ON/OFF

Mit TRACK LOOP kann man, bei Angabe von ON, die Musik, bzw. das Pattern endlos wiederholen. Wird das Ende erreicht, beginnt das Musikstück von neuem. Default ist OFF (=Aus).

Wichtige Informationen

Diese Befehle sind nicht als Ersatz für das eigentliche AMOS Music System gedacht. Sie ermöglichen zwar das Abspielen von Songs direkt von der Diskette, und umgehen so de mühsame Konvertierung, aber die AMOS Befehle zur Manipulation der Musik, wie TEMPO oder VOLUME wirken sich nicht auf Tracker Songs aus. Sie müssen die entsprechenden Werte schon bei dem Komponieren beachten. Es ist unmöglich, einen normalen AMOS Sound Befehl bei abspielender Tracker Musik zu benutzen. Dies wird zu Kollisionen führen. Eine Tracker Musik kann deshalb erst bei ausgeschalteter AMOs Musik abgespielt werden.

Das Spielen einer Note

PLAY *(Spielt eine Note)*

PLAY [Stimme,] Tonlage, Verzögerung

Spielt eine einzige Note über den Lautsprecher Ihres Fernsehers oder Stereoanlage. Tonlage bestimmt die Höhe dieses Tons; der Wert dafür liegt zwischen 0 (niedrig) und 96 (hoch). Es handelt sich hier jedoch nicht um willkürliche Nummern, sondern jede Tonlage ist mit einer der Noten in der Tonleiter verbunden (A, B, C, D, E, F, G). Die Zuweisung geht aus der folgenden Tabelle klar hervor.

	Oktave							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Note				**Tonlage**				
C	1	13	25	37	49	61	73	85
C#	2	14	26	38	50	61	74	86
D	3	15	27	39	51	62	75	87
D#	4	16	28	40	52	64	76	88
E	5	17	29	41	53	65	77	89
F	6	18	30	42	54	66	78	90
F#	7	19	31	43	55	67	79	91
G	8	20	32	44	56	68	80	92
G#	9	21	33	45	57	69	81	93
A	10	22	34	46	58	70	82	94
A#	11	23	35	47	59	71	83	95
B	12	24	36	48	60	72	84	96

Aus dieser Aufstellung wird deutlich, daß die Noten in Gruppen von 12 steigen. Diese Gruppen nennt man Oktaven.

Der Parameter *Stimme*, den Sie hier wahlweise eingeben können, erlaubt es Ihnen, Ihre Noten in jeder beliebigen Kombination der vier Stimmen des Amiga zu spielen. Wie üblich entspricht das einer Bitmap im folgenden Format:

Bit 0-> Stimme 0
Bit 1-> Stimme 1
Bit 2-> Stimme 2
Bit 3-> Stimme 3

Wenn Sie ein Bit auf 1 stellen, dann wird die Note über die entsprechende Stimme gespielt.

Verzögerung bestimmt die Länge der Pause zwischen dem PLAY-Befehl und der nächsten Basic-Anweisung. So können Sie immer erst eine Note spielen, bevor Sie zur nächsten gehen.

Wenn Sie für diese Verzögerung den Wert Null eingeben, so wird eine Note begonnen und sofort zur nächsten Basic-Anweisung gesprungen. Wenn Sie mehrere Noten nacheinander spielen, können Sie so ganz hübsche Harmonie- Effekte erzeugen. Hier sind einige Beispiele:

Play 1,40,0 : Play 2,50,0 : Rem Spielt ohne Verzögerung

```
        Wait Key
    Play 1,40,15 : Play 2,50,15 : Rem Beachte die Wirkung der Verzögerung
    Rem Spielt willkürliche Notenfolge
    Do
      T=Rnd(96) : V=Rnd(15) : Play V,T,3
    Loop
```

PLAY beschränkt sich übrigens nicht nur auf reine Noten. Sie können dem Sound-Generator auch durch die starken WAVE- und NOISE-Befehlen komplexe Wellenformen zuweisen.

Wellenformen und Hüllkurve

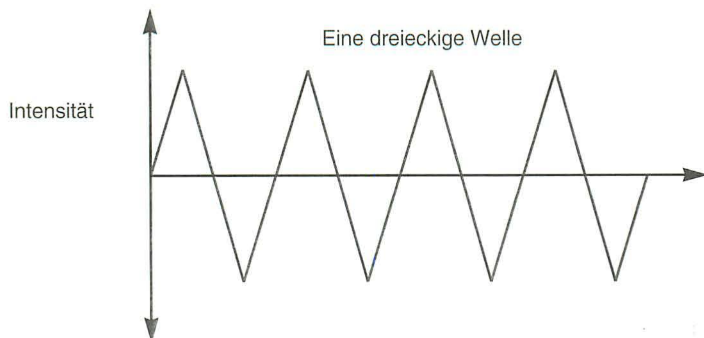
SET WAVE *(Definiert eine Wellenform)*

SET WAVE Welle,shape$

Die Anweisung SET WAVE bietet Ihnen die Möglichkeit, Ihre eigenen Instrumente zum Einsatz mit der PLAY-Anweisung in AMOS-Basic zu definieren. Der Klang Ihres Instruments hängt von dem Aussehen einer Wellenform ab, die sich im Speicher des Amiga befindet. Auf diese Art entsteht ein Template, das wiederholt wird, um schließlich die Endnote zu erzeugen.

Welle ist die Nummer der Wellenform, die Sie definieren möchten. Die zulässigen Wellennummern beginnen bei 2. Das kommt daher, weil die Wellen mit den Nummern 0 und 1 bereits installiert sind. Die Welle mit der Nummer Null enthält ein willkürliches Geräuschmuster für das Erzeugen von explosionsartigen Effekten. Die Welle eins ist eine glatte Sinus-Welle, die die reinen Töne erzeugt, die dann für die normale PLAY-Anweisung eingesetzt wird.

Wie Ihre Wellenformen aussehen, wird durch eine Reihe von 255 Werten bestimmt, die über den SHAPE$-Parameter eingegeben werden. Sehen Sie sich doch einmal das folgende Beispiel für eine solche Wellenform an:



Jeder Wert stellt die Intensität eines bestimmten Abschnittes der Wellenform dar. Das entspricht der Höhe eines einzigen Punktes in der obigen Grafik.

Die zulässigen Werte für die Intensität liegen zwischen -128 und 127. Da die AMOS-Zeichenketten nur **positive** Zahlen (0-255) enthalten können, müssen Sie Ihre negativen Werte vor dem Einsatz in ein besonderes internes Format umwandeln. Der entsprechende Wert kann einfach berechnet werden, indem Sie zu den negativen Werten in Ihrer Liste 256 addieren. So wird dann zum Beispiel -50 folgendermaßen eingegeben:

-50+256=206

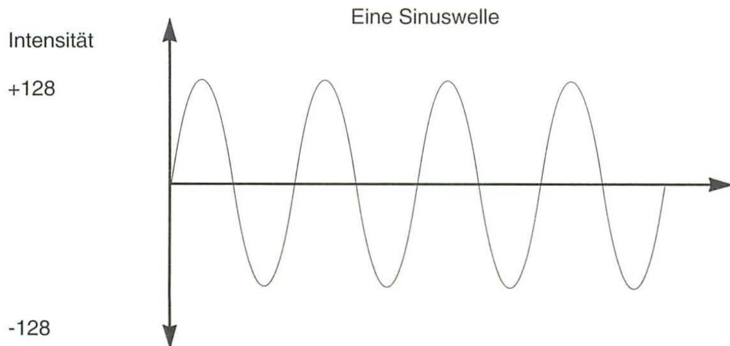
Hier ist ein Programm, das zeigt, wie die dreieckige Welle in dem vorstehenden Diagramm durch AMOS-Basic erzeugt werden kann.

```
S$="" : Rem Löscht Zeichenkette für Wellenform
For I=-128 To 127
   X=I : If X<0 Then Add X,256
   S$=S$+Chr$(X)
Next I
Set Wave 2,S$
```

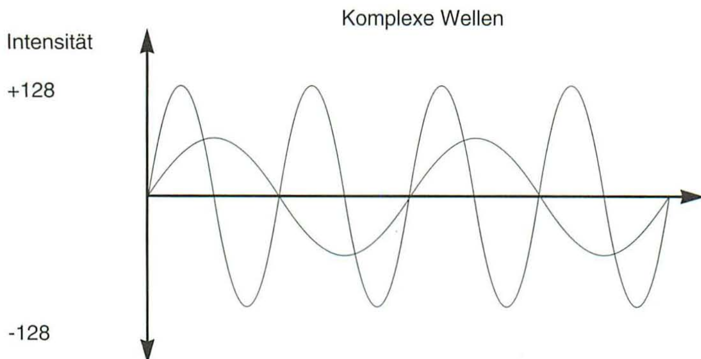
Bevor Sie Ihre Wellenform nun abspielen, müssen Sie AMOS-Basic mitteilen, welche Kanäle Ihrer Welle zugewiesen werden sollen. Dazu können Sie den WAVE- Befehl einsetzen. Fügen Sie zur oben angeführten Routine die folgende Zeile hinzu:

```
Wave 2 To 15 : For S=10 to 60 : Play S,10 : Next S
```

Am besten kann man den Klang eines echten Instrumentes reproduzieren, indem man verschiedene Sinuswellen miteinander verbindet. Ein Beispiel für solch eine Sinuswelle sehen Sie im folgenden Diagramm.



Wenn man mehrere dieser Wellen miteinander kombiniert, wobei sie jeweils unterschiedliche Größen und Anfangspunkte aufweisen sollten, dann kann man die folgenden Wellenmuster erzeugen.



Auf diese Art entstehen die glatten harmonischen Effekte, die Sie für Ihre Noten brauchen. Hier ist ein Beispiel dazu:

```
SHAPE$="" : Degree
For S=0 To 255
   V=Int((Sin(S)/2+Sin(S*2+45)/4)*128)+127
   SHAPE$=SHAPE$+Chr$(V)
Next S
Set Wave 2,SHAPE$ : Wave 2 To 15
For N=10 To 60 : Play N, 10 : Next N
```

WAVE *(Weist eine Welle einem oder mehreren Tonkanälen zu)*

WAVE w To v

WAVE weist die Welle mit der Nummer *w* einem oder mehreren Tonkanälen zu. Dabei enthält *v* eine Bitmap im Standardformat. Wenn ein Bit im dem Muster auf 1 gestellt wird, so werden die entsprechenden Stimmen von PLAY eingesetzt, andernfalls hat es auf sie überhaupt keine Auswirkungen.

Als Default wird die Welle Null dem NOISE-Kanal zugewiesen, und die Welle eins enthält eine Sinuswelle. Hier sind ein paar Beispiele:

```
Wave 0 To %0001 : Rem Weist Welle 0 der Stimme Null zu
Play 1,40,0
```

```
Wave 0 To %1100 : Rem Setzt Stimmen 3,2 für Geräusch ein
Play 20,10
Wave 1 To %1111 : Rem Spielt reinen Ton über alle vier Stimmen
Play 60,0
```

NOISE *(Weist eine Geräuschwelle einem Kanal zu)*

NOISE To Stimmen

NOISE weist den angegebenen Stimmen einen unbestimmten Geräuscheffekt (Welle Null) zu. Darauf baut denn eine große Vielzahl von Explosions- und Percussion-Effekten auf. Laden Sie **BEISPIEL 17.3** aus dem MANUAL- Unterverzeichnis und sehen Sie sich das anhand eines Beispiels an.

Stimmen ist ein normales Bitmuster. Die ersten vier Bits stellen die vier möglichen Stimmen des Amiga dar, angefangen bei Null. Wenn ein Bit auf eins gestellt wird, dann wird das Geräusch über diesen Kanal gespielt, andernfalls wird diese Stimme von der Anweisung überhaupt nicht verändert. NOISE entspricht dem Befehl:

```
Wave 0 To Stimmen
```

Beispiele:

```
NOISE To 15
Play 60,0
Play 60,0
```

DEL WAVE *(Löscht eine Welle)*

DEL WAVE n

Dieser Befehl löscht die Welle, die zuvor mit der Anweisung SET WAVE definiert wurde. *n* ist die Nummer der betreffenden Welle (die Numerierung beginnt bei 2). Es ist nicht möglich, die eingebauten NOISE- und SINUS-Wellen mit dieser Anweisung zu löschen. Nachdem die Welle gelöscht wurde, werden alle Stimmen wieder auf die normale Sinuswelle zurückgesetzt (Default).

SAMPLE *(Weist einer Welle ein Sample zu)*

SAMPLE n To Stimmen

Diese Ausführung ist die stärkste der WAVE-Anweisungen. Hier wird der aktuellen Welle ein Sample zugewiesen, das in der Sample-Bank gespeichert ist. Der Befehl PLAY greift jetzt direkt auf ein Instrument in der Sample-Bank zu.

```
Load "SAMPLES/SAMPLES.abk"
Sample 1 To 15
For I=20 To 50
   Play i,50
Next I
```

Wie üblich können Sie über *Stimmen* eine Reihe von Stimmen angeben, die von dieser Anweisung eingesetzt werden sollen. Das entspricht einer normalen Bitmap mit dem Format:

Bit 0-> Stimme 0
Bit 1-> Stimme 1
Bit 2-> Stimme 2
Bit 3-> Stimme 3

Jede Stimme kann wiederum dadurch ausgewählt werden, daß Sie das entsprechende Bit auf 1 stellen.

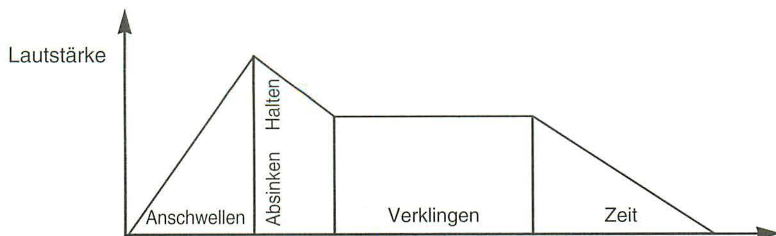
Achtung: Der Bereich der Noten mit dem ein Sample gespielt werden kann, hängt von der Geschwindigkeit ab, mit dem es ursprünglich aufgenommen wurde. Wenn eine Note zu hoch ist, dann kann AMOS sie möglicherweise überhaupt nicht spielen. Der zulässige Bereich ist von Sample zu Sample verschieden, liegt aber normalerweise zwischen 10 und 50.

SET ENVEL *(Erzeugt die Hüllkurve)*

SET ENVEL Welle,Phase TO Dauer,Lautstärke

Der Befehl SET ENVEL bewirkt eine fließende Änderung der Lautstärke einer Note während sie gespielt wird. In Wirklichkeit treten alle Töne nämlich nicht einfach fix und fertig auf. Sie zu bilden, dauert eine gewisse Zeit, und diesen Prozeß wird durch eine sogenannte Hüllkurve dargestellt. Wie diese Hüllkurve nun genau aussieht, hängt von der Art des Instruments ab, das Sie spielen. Hier ist jedoch ein typisches Beispiel für eine solche Kurve:

Eine typische Hüllkurve



Die Bildung dieses Tons erstreckt sich über vier Phasen: Anschwellen, Absinken, Halten und Verklingen.

Mit AMOS-Basic können Sie Ihre Hüllkurve in bis zu sieben verschiedene Schritte aufteilen. Jeder Schritt stellt eine konstante Veränderung der Lautstärke der aktuellen Note dar.

Welle ist die Nummer der Wellenform, auf die diese Anweisung wirkt. Sie können zu diesem Zweck jede beliebige Wellenform einsetzen, einschließlich der integrierten NOISE- und SINUS-Generatoren.

Phase enthält die Nummer der jeweiligen Phase, die definiert werden soll und liegt zwischen 0 und 6.

Dauer gibt die Dauer des aktuellen Schrittes in Einheiten von 50stel Sekunden an. Dadurch wird die Geschwindigkeit bestimmt, mit der die Änderung der Lautstärke in dieser Phase erfolgen soll.

Lautstärke gibt die Lautstärke an, die am Ende dieser Phase erreicht werden soll. Die zulässigen Werte liegen zwischen 0 und 63.

Dabei ist wesentlich, daß diese Lautstärke sich auf die Intensität bezieht, die Sie zuvor bereits mit dem Befehl VOLUME festgelegt haben. Wenn die Note also leise ist, bleibt trotzdem die Form der Hüllkurve vollkommen erhalten. Jetzt wollen wir uns einige Beispiele dazu ansehen:

```
Set Envel 1,0 To 200,63 : Rem Bestimmt die Länge des ersten Schrittes
Play 40,0 : Rem Diese Note wird vier Sekunden lang gespielt
```

Wie Sie hören können, beginnt die Lautstärke Ihres Tones bei Null, und steigert sich dann über die Dauer der Note bis zur maximalen Intensität. Nun wollen wir etwas versuchen, das ein bißchen komplizierter ist:

```
Set Envel 1,0 to 15,60 : Rem Langsames Anschwellen
Play 40,0 : Wait Key
Set Envel 1,1 To 1,50 : Rem Kurzes Absinken
Play 40,0 : Wait Key
Set Envel 1,2 To 10,50 : Rem Kurzes Halten
Play 40,0 : Wait Key
Set Envel 1,3 To 50,0 : Rem Langes Verklingen
Play 40,0
```

Und hier ist noch ein Beispiel für die Bildung eines Geräusches:

```
Noise To 15
Set Envel 0,0 To 1000,30
Play 40,0
Wait Key
Music Off : Rem Würgt Sound ab
```

Verwechseln Sie die Wellen und die Hüllkurve bitte nicht miteinander. Eine Welle bestimmt die Frequenzbestandteile Ihrer Noten und die Hüllkurve verändert nur einfach ihre Lautstärke nach einem bestimmten Muster.

Sprache

Ihr Amiga ist mit einem starken Programm zur Sprachumsetzung ausgestattet, das sich auf der normalen Workbench-Diskette befindet. Mit Hilfe dieser Routine können Sie Ihren AMOS-Programmen das Sprechen beibringen. Der Einsatz von Sprache ist besonders bei Lernprogrammen wichtig, denn viele junge Leute sprechen wesentlich besser auf das gesprochene Wort als auf einen langweiligen Text an.

Hier jedoch ein Wort der Warnung. Da das NARRATOR-Programm von AMOS-Basic ganz unabhängig ist, können wir nicht für seine Zuverlässigkeit bürgen. Sie werden wahrscheinlich keine ersten Schwierigkeiten damit haben, aber sollten trotzdem mit etwas Vorsicht ans Werk gehen.

SAY *(Spricht einen Satz)*

SAY t$[,Modus]

Der Befehl SAY ist unwahrscheinlich leicht anzuwenden. Geben Sie einfach Ihren Text in normalem Englisch ein und schließen Sie Ihren Satz mit einem Satzzeichen wie zum Beispiel einem Punkt ab. Dann setzt SAY Ihre Worte in ein internes Format um und sie ertönen umgehend aus Ihrem Lautsprecher. Hier ist ein Beispiel dazu:

```
Say "AMOS-Basic kann wirklich sprechen."
```

Wenn Sie diese Anweisung zum ersten Mal einsetzen, dann wird das NARRATOR-Werkzeug automatisch von der Diskette geladen. Sie müssen also unbedingt sicherstellen, daß die entsprechende Diskette im aktuellen Laufwerk ist, bevor Sie dieses System einsetzen, sonst erhalten Sie eine Art Intuition-Requester-Kasten.

Modus schaltet zwischen den beiden verschiedenen Sprachmodi hin und her. Als Default wartet Ihr Programm ab, bis der Spracheffekt abgeschlossen ist und unterbricht in dieser Zeit alle Musik- und Sound-Effekte. Wenn Sie jedoch für *Modus* den Wert 1 einstellen, so wird ein Multitasking-System aktiviert, und Sie können auch Sprache ausgeben, während AMOS Ihr Programm ablaufen läßt. Das führt natürlich zwangsläufig dazu, daß Ihre Basic-Routinen erheblich verlangsamt werden. Um Ihre Sprache auf normal zurückzusetzen, stellen Sie einfach *Modus* auf Null. Dazu folgendes Beispiel:

```
Do
   Input "Satz eingeben?";T$
   T$=T$+"."
   Say T$,1
Loop
```

Wenn das Narrator-System nicht versteht, was Sie sagen möchten, dann erhalten Sie keine Fehlermeldung, sondern der Befehl wird automatisch abgebrochen.

Beachten Sie in diesem Zusammenhang auch, daß das Narrator manchmal durch ganz kurze Sätze ein bißchen durcheinander gebracht werden kann. Manchmal wird der Rest des vorhergehenden Satzes an den aktuellen angehängt. Dieses Problem können Sie lösen, indem Sie einfach eine Reihe von Leerschritten an das Ende Ihres Textes anhängen. So werden dann unerwünschte Sprachdaten gelöscht.

SET TALK *(Einsatz von Spracheffekten)*

SET TALK Geschlecht,Modus,Tonlage,Geschwindigkeit

SET TALK ermöglicht es Ihnen, die Art der Stimme, die für die Ausführung des SAY-Befehls verwendet wird, zu verändern.

Geschlecht wählt zwischen *männlicher* (0) und *weiblicher* (1) Stimme. Um ganz ehrlich zu sein, das Ergebnis klingt in keinem der beiden Fälle besonders realistisch. Sie können eigentlich einfach durch das Erhöhen der Stimmenfrequenz durch den Parameter *Tonlage* bessere Effekte erzielen.

Modus gibt der Stimme einen eigenartigen Sprachrhythmus. Sie können ihn einfach dadurch aktivieren, daß Sie *Modus* auf den Wert 1 setzen. Das hat eine ganz angenehme Wirkung, klingt aber trotzdem nicht unbedingt menschlich. Wenn Sie diesen Parameter auf Null stellen, dann wird die Stimme wieder auf normal zurückgesetzt.

Tonlage verändert die Frequenz der Stimme von 65 (niedrig) bis 320 (unwahrscheinlich hoch).

Geschwindigkeit gibt die Geschwindigkeit in der Anzahl der Wörter pro Minute an. Die zulässigen Werte liegen hier zwischen 40 und 400.

Jeder der obengenannten Parameter kann auch weggelassen werden. Vorausgesetzt Sie lassen die Kommas an der üblichen Stelle, können Sie die Optionen beliebig kombinieren. Hier sind einige Beispiele:

```
Set Talk 1,,,: Say "Das ist eine Frauenstimme."
Set Talk ,1,,: Say "Das ist eine Frauenstimme mit Betonung."
Set Talk 0,0,320,: Say "Eine Sopran-Stimme."
Set Talk ,,65,: Say "Eine tiefe Stimme."
Set Talk 0,0,100,400 : Say "Sagt das ganz schnell."
Set Talk ,,,40 : Say "Das ist langsam."
```

Filter-Effekte

LED *(Aktiviert einen Hochpaßfilter/verändert POWER-LED)*

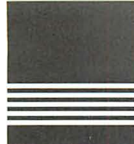
LED ON/OFF

Der LED-Befehl hat zwei völlig getrennte Auswirkungen. Einmal schaltet er die POWER-LED an Ihrer Amiga-Konsole aus und an, und zum anderen steuert er aber auch einen besonderen Hochpaßfilter.

Dieser Filter verändert die Art, auf die Töne mit hoher Frequenz von dem System behandelt werden. Normalerweise werden diese Töne herausgefiltert, damit das Risiko von unerwünschten Verzerrungen vermieden werden kann. Unglücklicherweise nimmt das vielen Schlaginstrumenten das Timbre. Wenn Sie also den Filter ausschalten, können Sie die eigentliche Qualität vieler Instrumente wieder erfassen. Dazu folgendes Beispiel:

```
Load "AMOS_DATA:MUSIC/MUSICDEMO.ABK" : Rem von der AMOS Datendiskette.
Music 1
Do
   If mouse key=1 Then Led on
   If mouse key=2 Then Led off
Loop
```

Wenn Sie mit dieser Funktion etwas experimentieren, werden Sie feststellen, daß manche Musikstücke ohne den Filter wesentlich besser klingen, andere hingegen scheußlich verzerrt werden, wenn der Filter ausgeschaltet ist. Durch die POWER- LED können Sie jetzt auf einen Blick sehen, wie der Filter eingestellt ist. Wenn Sie ihn im Tempo der Musik bewegen, dann können Sie den Sound damit ganz toll steigern.



18: Die Tastatur

Irgendwann in der fernen Zukunft werden wir unsere Programme direkt durch eine Art von Gedankenübertragung eingeben. Bis dahin bleibt jedoch das Eintippen von Information über die Tastatur nach wie vor die schnellste Methode der Dateneingabe in den Computer.

AMOS-Basic bietet Ihnen unzählige nützliche Tastaturbefehle. Sie können ganz universell, vom Ballerspiel bis zum Abenteuer eingesetzt werden. Sie können sogar ausschließlich in AMOS-Basic ein vollständiges Textverarbeitungsprogramm schreiben!

=INKEY$ *(Fragt Tastendruck ab)*

k$=INKEY$

Die INKEY$-Funktion fragt ab, ob der Anwender eine Taste gedrückt hat und weist den Wert dieser Taste dann in der Zeichenkette k$ aus.

Beachten Sie dabei, daß der INKEY$-Befehl den Input überhaupt nicht abwartet. Wenn der Anwender kein Zeichen eingegeben hat, liefert INKEY$ einfach eine leere Kette "". Hier ist ein Beispiel dazu:

```
Do
   Rem Versuch ein paar Tasten zu drücken
   X$=Inkey$: If X$<>"" Then Print X$;
Loop
```

INKEY$ kann nur Tasten abfragen, die ein bestimmtes ASCII-Zeichen ergeben. ASCII ist ein Standardcode, der alle Zeichen, die auf dem Bildschirm erscheinen können, darstellt.

Dabei ist wichtig, daß einige der Tasten, wie zum Beispiel die Hilfe-Taste oder die Funktionstasten ein ganz anderes Format verwenden. Wenn INKEY$ solch eine Taste ermittelt, dann wird ein Zeichen mit dem Wert Null (CHR$(0)) ausgewiesen. Sie können jetzt den internen Scan-Code dieser Taste mit der besonderen SCAN CODE Funktion abfragen.

=SCANCODE *(Gibt den Scan-Code der letzten mit INKEY$ eingegebenen Taste an)*

s=SCANCODE

SCANCODE weist den internen Scan-Code einer Taste aus, die zuvor über die INKEY$-Funktion eingegeben wurde. So können Sie die Tasten abfragen, die keine Zeichen der Tastatur erzeugen, wie zum Beispiel die HILFE- oder TAB-Tasten. Versuchen Sie es doch einmal mit einem kleinen Beispiel:

```
Do
  While K$=""
      K$=Inkey$
  Wend
  If Asc(K$)=0 Then Print "Du hast eine Taste ohne ASCII-Code gedrückt."
  Print "Der Scancode ist";Scancode
  K$=""
Loop
```

=KEY STATE *(Abfrage, ob eine bestimmte Taste gedrückt wurde)*

t=KEY STATE(s)

Überprüft, ob eine bestimmte Taste auf der Amiga-Tastatur gedrückt wurde. s ist der interne Scan-Code der Taste, die Sie abfragen möchten. Wenn diese Taste gerade gedrückt wird, so weist KEY STATE den Wert -1 (richtig - true) aus, andernfalls lautet das Ergebnis 0 (falsch - false). Hier ist ein Beispiel:

```
Do
  If Key State(69)=True Then Print "Ausgestiegen!" : Rem Esc-Taste gedrückt
  If Key State(95)=True Then Print "HILFE!" : Rem Hilfe-Taste gedrückt
Loop
```

=KEY SHIFT *(Weist den Status der Shift-Tasten aus)*

Tasten=KEY SHIFT

KEY SHIFT gibt den aktuellen Status der verschiedenen Steuertasten aus. Diese Tasten, wie SHIFT und ALT zum Beispiel, können mit dem normalen INKEY$- oder SCANCODE-System nicht abgefragt werden. Aber durch das Aufrufen der KEY SHIFT Funktion können Sie ganz leicht jede beliebige Kombination dieser Steuertasten überprüfen. *Tasten* stellt eine Bitmap mit folgenden Format dar:

Bit	Abgefragte Taste	Bemerkungen
0	Linke SHIFT-Taste	
1	Rechte SHIFT-Taste	
2	CAPS LOCK	Entweder AN oder AUS
3	Cntrl	
4	Alt links	
5	Alt rechts	
6	Linke Amiga-Taste	Auf manchen Tastaturen ist das die Commodore-Taste
7	Rechte Amiga-Taste	

Wenn ein Bit auf 1 gestellt wurde, dann wurde die entsprechende Taste vom Anwender gedrückt. Dazu folgendes Beispiel:

```
Centre "<Drück einige Steuertasten>"
Curs Off
Do
  Locate 14,4 : Print Bin$(Key Shift,8)
Loop
```

=INPUT$(n) *(Funktion zur Eingabe von n Zeichen in eine Zeichenkette)*

x$=INPUT(n)

INPUT$ gibt *n* Zeichen direkt über die Tastatur ein und wartet dabei jedes einzelne ab. Wie bei INKEY$ werden diese Zeichen nicht auf dem Bildschirm angezeigt.

x$ ist eine Zeichenketten-Variable, in die Ihre neuen Zeichen geladen werden. *n* gibt die Anzahl der Zeichen, die eingegeben werden sollen, an. Dazu folgendes Beispiel:

```
Clear Key : Print "Tippe zehn Zeichen ein"
C$=Input$(10) : Print "Du hast eingegeben ";C$
```

Diese Anweisung entspricht dem normalen INPUT-Befehl **nicht**. Die beiden Anweisungen sind völlig verschieden. Beachten Sie auch, daß es eine spezielle Version von INPUT$ gibt, mit der Sie Ihre Zeichen auf der Diskette lesen können.

WAIT KEY *(Ein Tastendruck wird abgewartet)*

WAIT KEY

Wartet einen Tastendruck ab. Hier ist ein Beispiel dazu:

```
Print "Drück eine Taste" : Wait Key : Print "Taste gedrückt"
```

KEY SPEED *(Verändert die Anschlaggeschwindigkeit)*

KEY SPEED Verschiebung,Geschwindigkeit

Mit KEY SPEED können Sie sich die Geschwindigkeit der Tastatur ganz nach Ihrem eigenen Geschmack einstellen. Die neue Geschwindigkeit wird dann für alle Teile des AMOS-Systems angewendet, einschließlich des Editors.

Verschiebung ist die Zeitspanne (in 50stel Sekunden), die zwischen dem Drücken einer Taste und der Wiederholungssequenz liegen soll.

Geschwindigkeit ist die Verzögerung (in 50stel Sekunden), die zwischen jedem einzelnen Zeichen erfolgen soll. Dazu folgendes Beispiel:

```
key$(1)="Tastengeschwindigkeit 10,1"+Chr$(13) : Rem Speichert vernünftige
Geschwindigkeit
Key Speed 10,10 : Rem Taste gedrückt halten für LANGSAME Wiederholung
Key Speed 1,1 : Rem SCHNELLE Wiederholung
Rem Drück linke Amiga F1 Taste, dann arbeitet Tastatur wieder normal
```

CLEAR KEY *(Initialisiert Tastaturpuffer)*

CLEAR KEY

Jedesmal, wenn Sie ein Zeichen über die Tastatur eingeben, wird sein ASCII-Code in einen Speicherbereich gestellt, den man Tastaturpuffer nennt. Und auf diesen Puffer greift die INKEY$-Funktion dann zu, um die Tastaturabfrage durchzuführen.

CLEAR KEY löscht diesen Puffer vollständig, und bringt Ihre Tastatur in den Ausgangszustand zurück. Das ist vor allem zu Beginn eines Programm sehr nützlich, denn der Puffer kann noch voll von unerwünschten Informationen sein. Sie können diese Anweisung also unmittelbar vor einem WAIT KEY Befehl aufrufen, um sicherzugehen, daß das Programm erst einen neuen Tastendruck abwartet, bevor es weitergeht. Ein Beispiel:

```
Clear Key : Rem Löscht den aktuellen Tastendruck
Wait Key : Rem Wartet auf neue Taste
```

PUT KEY *(Stellt eine Zeichenkette in den Tastaturpuffer)*

PUT KEY a$

Lädt eine Zeichenkette direkt in den Tastaturpuffer. Auch Zeilenschaltungen können durch die Verwendung des Zeichens CHR$(13) (ENTER) einbezogen werden.

Meistens wird PUT KEY eingesetzt, um Defaults für Ihre INPUT-Routinen einzugeben. Hier ist eine Demonstration:

```
Do
  Put Key "Nein"
  Input "Noch ein Spiel";A$
  If A$="Nein" Then Exit
Loop
```

Das oben aufgeführte Programm weist der Default-INPUT-Kette "Nein" zu. Das Drücken der ENTER-Taste gibt diesen Wert direkt in die Variable A$ ein. Alternativ dazu können Sie die Zeile auch über die normalen Cursortasten bearbeiten und Ihren eigenen Wert für A$ je nach Wunsch eintippen.

Input/Output

INPUT *(Lädt einen Wert und stellt ihn in eine Variable)*

INPUT bietet Ihnen zwei Standardmöglichkeiten für die Eingabe von Informationen in eine oder mehrere Variablen. Für diese Anweisung gibt es zwei mögliche Formate:

INPUT vars[;]

Gibt eine Reihe von Variablen direkt über die Tastatur ein. var kann jeden beliebigen Satz von Variablen enthalten, die voneinander durch Kommas abgetrennt sind. Es erscheint automatisch ein Fragezeichen an der aktuellen Cursor-Position.

INPUT "Prompt";Variablenliste[;]

Zeigt die *prompt*-Zeichenkette an, bevor Ihre Information eingegeben wird. Beachten Sie, daß Sie zwischen Ihrem Text und der Variablenliste immer einen Semikolon einfügen müssen. Sie können zu diesem Zweck **kein** Komma einsetzen.

Am Ende Ihrer Variablenliste können Sie wahlweise einen Semikolon setzen, das gibt dann an, daß der Text-Cursor von der INPUT-Anweisung nicht betroffen sein soll und seine ursprüngliche Position beibehält, wenn die Daten eingegeben sind.

Wenn Sie einen dieser Befehle ausführen, so wartet das Basic darauf, daß Sie die entsprechenden Informationen über die Tastatur eingeben. Jede Variable in Ihrer Liste muß von einem durch den Anwender eingegebenen Wert begleitet werden. Diese Werte müssen vom selben Typ sein wie Ihre ursprünglichen Variablen und sollten durch Kommas abgetrennt werden. Hier sind ein paar einfache Beispiele:

```
Print "Du hast eingegeben";N#
Input "Wie heißt Du, Mensch?";name$;
Locate 23, : Print "Hallo ";Name$
```

Siehe auch INPUT# und LINE INPUT.

LINE INPUT *(Eingabe einer Reihe von Variablen, die durch ENTER voneinander abgetrennt werden)*

INPUT "Prompt";Variablenliste[;]
INPUT vars[;]

Line Input entspricht INPUT genau, davon abgesehen, daß hier ein ENTER (Zeilenschaltung) statt eines Kommas dazu eingesetzt wird, um die Werte, die Sie über die Tastatur eingeben, voneinander abzutrennen. Dazu folgendes Beispiel:

```
Line Input "Gib drei Zahlen ein";A,B,C
Print A,B,C
```

Siehe auch INPUT, LINE INPUT#.



19: Sonstige Befehle

Wie alle Versionen der Basic-Programmiersprache enthält auch AMOS eine Vielzahl von so alltäglichen Befehlen wie PRINT und DATA. In diesem Kapitel werden Sie feststellen, daß AMOS auch hier eine Reihe von interessanten neuen Möglichkeiten für diese Anweisungen bietet. Und so werden auch die etwas langweiligeren Teile von AMOS-Basic noch richtig spannend!

PRINT or ? *(Zeigt eine Reihe von Variablen auf dem Bildschirm an)*

PRINT items

Durch die PRINT-Anweisung werden Informationen auf dem Bildschirm angezeigt, angefangen bei der aktuellen Cursor-Position.

Diese Reihe von Daten kann aus jeder beliebigen Gruppe von Variablen oder Konstanten bestehen, vorausgesetzt sie überschreitet die zulässige Höchstlänge von 255 Zeichen nicht.

Alle Elemente dieser Reihe müssen voneinander entweder durch einen Strichpunkt ";" oder ein Komma "," abgetrennt werden. Wenn Sie dafür einen Strichpunkt einsetzen, so werden die Daten unmittelbar nach dem nächsten Wert erscheinen; bei einem Komma hingegen springt der Cursor erst zum nächsten Tabulator auf dem Bildschirm.

Normalerweise wird der Cursor nach jeder PRINT-Anweisung eine Zeile nach unten verschoben. Das kann aber durch das Einfügen eines Trennungszeichens unterdrückt werden. Auch durch einen Strichpunkt wird die Cursor-Position auf dem Bildschirm beibehalten, und durch ein Komma wird der Cursor zur nächsten TAB-Position verschoben.

```
Print "Per Anhalter durch die Galaxis."
A=10 : B=20 : C$="Dreißig" : Print A,B;C$
Print 10,20*10,"Hal";
Print "lo"
```

Siehe dazu auch die Befehle USING, LPRINT und PRINT#.

USING *(Formatierte Ausgabe)*

PRINT USING format$;variable list

Die Anweisung USING wird in Verbindung mit PRINT eingesetzt, um das Ausgabeformat ganz genau zu bestimmen.

format bestimmt eine Reihe von Zeichen, die wiederum definieren, wie Ihre Variablen auf dem Bildschirm dargestellt werden. Der ganze normale Text in dieser Zeichenkette wird direkt angezeigt, aber wenn Sie eines der Zeichen ~#+-.;^

miteinbeziehen, dann können Sie eine der folgenden, nützlichen Formatierungsmöglichkeiten einsetzen.

~ (Shift+#) Formatiert eine Zeichenkettenvariable. Jedes ~ wird von links nach rechts durch ein Zeichen Ihrer Ausgabekette ersetzt.

Print Using "Das ist eine ~~~~~~ Demonstration von USING";"kleine"
Das ist eine kleine Demonstration von USING

Jede Raute stellt eine Stelle Ihrer Variablen dar, die angezeigt werden soll. Alle nicht eingesetzten Stellen in dieser Auflistung werden automatisch durch Leerschritte ersetzt.

Print Using "####";314211
4211

+ Fügt zu einer positiven Zahl ein Pluszeichen und zu einer negativen Zahl ein Minuszeichen hinzu.

Print Using "+##";10: Print Using "+##";-10
+10
-10

- Weist nur dann ein Zeichen aus, wenn die Zahl negativ ist. Vor positiven Zahlen wird ein Leerschritt gesetzt.

Print Using "-##";10: Print Using "-##";-10
 10
-10

. Ein Punkt fügt in eine Zahl einen Dezimalpunkt ein und zentriert sie dann sauber auf dem Bildschirm.

Print Using "PI ist #.###";3.1415926
PI ist 3.141

; Zentriert eine Zahl, fügt aber keinen Dezimalpunkt ein.

Print Using "PI ist #;###";PI#
PI ist 3 141

^ (Shift 6) stellt eine Zahl im Exponentialformat dar.

Print Using " Hier ist eine Zahl ^";12345.678
Hier ist eine Zahl 1.2345678E5

REM or *(Bemerkung)*

REM Bemerkung

Die REM-Anweisung wird dazu verwendet, um zu Ihrem Basic-Programm Bemerkungen hinzuzufügen. Der gesamte Text, den Sie nach einer Rem-Anweisung eintippen, wird von AMOS-Basic völlig ignoriert. Hier ist ein Beispiel:

```
Rem Dieses Programm tut überhaupt nichts
` Dies ist eine Bemerkung
```

Diese normale Rem-Anweisung kann so gut wie überall in Ihrem Basic-Programm eingesetzt werden. Aber Sie können nur ganz am Anfang einer Zeile ein `-Anführungszeichen setzen. Dazu einige Beispiele:

```
`Eine einfache Bemerkung
Print "Mist" : Rem Das ist ok
Print "Goodbye" : ' Das ruft eine Fehlermeldung hervor
```

DATA *(Fügt eine Reihe von Daten in ein AMOS Basic-Programm ein)*

DATA Reihe von Daten

Über die DATA-Anweisung können Sie ganze Listen von nützlichen Informationen direkt in ein Basic-Programm einfügen. Diese Daten können dann in der Folge mit der READ-Anweisung in eine oder mehrere Variablen geladen werden. Alle Variablen in Ihrer Auflistung werden durch ein Komma voneinander abgetrennt. Hier ist ein Beispiel dazu:

```
Data 1,2,3"Hallo"
```

Anders als bei den meisten anderen Basic-Versionen, können Sie in AMOS bei dieser Anweisung auch Ausdrücke als Teil Ihrer Daten miteinbeziehen. Die folgenden Codezeilen sind also alle zulässig:

```
Data $FF50,$890
Data %1111111111111,%1101010101
Data A
Label: Data A+3/2.0-Sin(B)
Data "Hallo"+"Du"
```

Dabei ist wichtig, daß das "A" bei LABEL als der Inhalt der Variablen A und nicht als das Zeichen A eingegeben wird. Der Ausdruck wird dann automatisch während der READ-Operation berechnet, dabei werden die neuesten Werte für A und B verwendet.

Beachten Sie auch, daß jede DATA-Anweisung der einzige Befehl in der jeweiligen Zeile sein muß. Alles, was nach diesem Befehl steht, wird nämlich völlig ignoriert! DATA-Anweisungen können an jeder beliebigen Stelle in Ihren Basic-Programmen eingefügt werden. Jede Prozedur kann über ihren eigenen Satz von DATA- Anweisungen verfügen, die dann völlig unabhängig vom Rest Ihres Programmes sind. Das wollen wir anhand des folgenden Beispiels demonstrieren:

```
TEST : Read A$ : Print A$
Data "Programmdaten"
Procedure Test
   Read B$ : Print B$
   Data "Prozedurdaten"
End proc
```

Siehe auch die Befehle READ, RESTORE.

READ (Lädt Daten aus einer DATA-Anweisung in eine Variable)

READ Variablenliste

READ lädt Informationen, die in einer DATA-Anweisung gespeichert sind, in eine Reihe von Variablen. READ verwendet dazu eine spezielle Markierung, um den Ort zu bestimmen, an dem der nächste Teil der Daten gelesen werden soll. Am Anfang Ihres Programms wird diese Markierung auf das erste Element in der ersten DATA-Anweisung gesetzt. Wenn dieses Element dann gelesen wurde, wird die Markierung weitergeschoben, so daß sie auf das nächste Element in Ihrer Liste weist. Und das haben Sie wahrscheinlich auch schon erwartet: der Typ der Variablen, die Sie mit dieser Anweisung lesen, muß genau mit dem Typ der Daten, die sich an der aktuellen Position befinden, übereinstimmen. Hier ist ein Beispiel dazu:

```
T=10
Read A$,B,C,D$
Print A$,B,C,D$
Data "String",2,T*20+rnd(100),"AMOS"+"Basic"
```

Siehe auch RESTORE, DATA.

RESTORE (Setzt den aktuellen READ-Zeiger)

RESTORE Label
RESTORE LABEL$
RESTORE Zeile
RESTORE Nummer

Durch RESTORE können Sie den Punkt verändern, an dem eine nachfolgende READ-Operation die nächste DATA-Anweisung erwartet. Jede AMOS Prozedur verfügt über ihren eigenen DATA-Zeiger. Deshalb wirkt dieser Befehl immer nur auf die **aktuelle** Prozedur.

Label ist eine Sprungmarke, die die Position der ersten DATA-Anweisung angibt, die gelesen werden soll. Die Bezeichnung dieser Sprungmarke kann als Teil eines mathematischen Ausdrucks auch berechnet werden. Die folgenden Basic-Befehle sind also alle völlig in Ordnung:

```
Restore L
Restore "L"+"A"+"B"+"E"+"L"
```

Entsprechend geben Sie über den Parameter *Zeile* die Zeile der nächsten DATA-Anweisung ein. Wie bei *Label* kann auch hier ein Ausdruck eingeben werden:

```
Restore 10
Restore TEST+2
```

RESTORE erlaubt es Ihnen, beliebig durch die DATA-Anweisungen in Ihrem Programm zu springen. So können Sie Ihre Informationen dem Verhalten des Anwenders anpassen. Sie können zum Beispiel bei einem Abenteuerspiel die Beschreibung jedes Raumes in einer Reihe von einfachen DATA-Anweisungen speichern. Diese Beschreibung könnten Sie dann zum Beispiel mit den folgenden Zeilen lesen lassen:

```
Restore RAUM*5+1000 : Rem Jeder RAUM hat 5 DATA-Anweisungen
Read DESC$ : Print DESC$
1000 Data "Beschreibung von Raum 1"
1005 Data "Text Raum 2"
1010 Data "Raum 3"
:           :           :
```

Wenn natürlich in der Zeile, die Sie mit RESTORE angeben, keine DATA-Anweisung ist, erhalten Sie eine entsprechende Fehlermeldung. Es ist nicht ratsam, diesen Befehl innerhalb einer Prozedur einzusetzen. Damit die Sache funktioniert, **müssen** Ihre DATA-Anweisungen in der aktuellen Prozedur enthalten sein.

Siehe auch READ, DATA.

WAIT *(Wartet eine 50stel Sekunde)*

WAIT n

Unterbricht ein AMOS Basic-Programm für n 50stel Sekunden. Alle Funktionen, die Interrupts einsetzen wie zum Beispiel MOVE und MUSIC, funktionieren jedoch während

dieser Zeit wie üblich. Hier ist ein Beispiel dazu:

Wait 50

Jetzt wird eine Sekunde lang gewartet.

=TIMER= *(Zählt in 50stel Sekunden)*

V=TIMER
TIMER=v

TIMER ist eine reservierte Variable, die bei jeder 50stel Sekunde um eins zunimmt. Sie wird normalerweise dafür verwendet, die Basis für das Generieren der Zufallszahlen festzulegen. Dazu folgendes Beispiel:

Randomize Timer

MULTI WAIT *(Erzwingt ein Wait-VBL für Multi-Tasking)*

MULTI WAIT

Um echte Multi-Tasking-Programme zu erzeugen, dürfen Sie nicht die gesamte Prozessorzeit horten, sondern Sie sollten auch für die anderen Programmteile etwas Leistung übrig lassen. Der Befehl MULTI WAIT bewirkt ein MULTI-TASK Wait VBL (Vertikal Blank). Sie sollten diesen Befehl in der Hauptschleife Ihres Programms einsetzen, wenn Sie zum Beispiel darauf warten, daß ein Menüpunkt gewählt wird.

Beachten Sie dabei aber, daß Sie mit dieser Anweisung nicht versuchen sollten, eine genaue Bildschirmsynchronisation zu erreichen. Da sie für das Multi-Tasking bestimmt ist, ist diese Anweisung überhaupt **nicht** konsistent! Sie kann je nach Anzahl der im Moment laufenden Tasks eine ganze Reihe von VBLs überspringen.

Das Multi-Tasking kann übrigens durch Drücken von Amiga+A aktiviert werden. Dann kann man zwischen AMOS und dem CLI- oder Workbench-Umfeld hin- und herspringen. Auf diese Art können auf Systemen mit mindestens 1 MegaByte AMOS und Programme wie DPAINT III gleichzeitig laufen!

NOT *(Logische NOT-Operation)*

V=NOT(d)

Diese Funktion verändert jede Binärstelle in einer Zahl von 1 auf 0 und umgekehrt. Da Richtig (True) =-1 (%1111111111111) in Binärschreibweise und Falsch (False) =0 ist, ist NOT(True)=False. Hier ist ein Beispiel dafür:

```
Print Bin$(Not(%1010),4)
0101
If Not(True)=False Then Print "False"
```

TRUE *(Logisches RICHTIG)*

v=TRUE

Jedesmal, wenn eine Abfrage wie zum Beispiel X>10 durchgeführt wird, erhält man einen Wert. Wenn die Bedingung zutrifft (richtig = true ist), dann wird dieser Wert auf -1 gestellt, andernfalls lautet er 0.

```
If -1 Then Print "Minus 1 Is TRUE"
If TRUE Then Print "And TRUE Is";TRUE
```

Siehe auch FALSE und NOT.

FALSE *(Logisches FALSCH)*

v=FALSE

Weist den Wert Null aus. Dieser Wert zeigt bei allen Anweisungen, die eine Bedingung enthalten, wie IF...THEN oder REPEAT...UNTIL an, daß das Ergebnis der Abfrage FALSCH (False) lautet.

```
Print FALSE
0
```

Siehe auch TRUE.

AMOS TO BACK *(Schiebt AMOS aus dem Blickfeld und zeigt die Workbench)*

AMOS TO BACK

Durch diesen Befehl wird die Workbench-Anzeige in den Vordergrund gerückt, so daß Sie Zugang zu anderen Programmen erhalten.

AMOS TO FRONT *(Bringt AMOS auf die Anzeige)*

AMOS TO FRONT

Durch diesen Befehl wird AMOS wieder in den Vordergrund der Anzeige gebracht und die Workbench verborgen.

AMOS HERE
(Abfrage, welches Programm angezeigt wird)

AMOS HERE

Es wird TRUE (= Richtig) ausgewiesen, wenn AMOS gerade angezeigt wird und FALSE (= Falsch) angegeben, wenn die Workbench im Blickfeld ist.

AMOS LOCK
(Schaltet Umschaltmöglichkeit ab)

AMOS LOCK

Dieser Befehl führt zunächst das AMOS TO FRONT Kommando aus und verhindert dann die Umschaltmöglichkeit zur Workbench mit Amiga+A. Benutzen Sie diesen Befehl z.B. um nicht gleich offensichtlich zu machen, daß Ihr Programm mit AMOS geschrieben wurde.

AMOS UNLOCK
(Aktiviert Umschaltmöglichkeit)

Ermöglicht eine zuvor mit AMOS LOCK verbotene Umschaltmöglichkeit zur Workbench. Nach dem Aufruf von AMOS LOCK kann der Benutzer wieder mittels Amiga+A hin- und herschalten.

PRG STATE
(Gibt den Status des aktuellen Programmes aus)

PRG STATE gibt den Status des gerade ablaufenden Programmes aus. Es kann dabei 3 verschiedene Werte zurückgeben :

1 : Wenn das Programm unter dem Interpreter läuft.
0 : Wenn nur das Run Time-System genutzt wird.
-1 : Wenn es sich um ein kompiliertes Programm handelt.

20: Der Zugriff auf den Speicher

Über die Disk-Befehle in AMOS-Basic erhalten Sie uneingeschränkten Zugang zum Speichersystem des Amiga. Sie können damit so ziemlich alles machen, vom einfachen Lesen der Dateien bis hin zur Erstellung einer richtigen Datenbank.

Besonders beeindruckend ist AMOS jedoch vor allem, was die Auswahl der Dateien betrifft. Es gibt eine eingebaute Datei-Selektor-Routine, durch die Sie Ihre Dateinamen über eine raffinierte Dialogbox aussuchen können. Dieser Datei- Selektor ist ganz einfach zu benutzen und läßt Ihre Basic-Programme noch professioneller erscheinen.

Darüber hinaus kann AMOS Verzeichnisse (Directories) lesen, Dateien löschen oder direkt aus einem Ihrer Programme heraus Unterverzeichnisse erstellen. Es gibt sogar einen Befehl, mit dem Sie nach einer bestimmten Datei auf der Diskette suchen können.

Und zu guter Letzt können wir Sie noch besonders unterstützen, wenn Sie vom ursprünglichen STOS Basic-Paket zu AMOS aufrüsten. AMOS ist völlig zu dem leistungsstarken CROSS-DOS-System von CONSULTRON kompatibel. Wenn Sie also dieses Produkt gekauft haben, können Sie Ihre vorhandenen STOS Basic- Programme ganz leicht direkt in AMOS-Basic importieren.

Laufwerke und Volumes

Wie Sie wissen, können Sie mit AMOS Ihre Disketten auf ein paar verschiedene Arten bezeichnen. Wenn Sie mit CLI nicht so gut vertraut sind, so finden Sie vielleicht manche diese Bezeichnungen etwas verwirrend. Deshalb wollen wir Ihnen eine kurze Erklärung zu den verschiedenen Bezeichnungsarten geben.

Laufwerke

Jedes an den Amiga angeschlossene Laufwerk wird mit dem üblichen, aus drei Buchstaben bestehenden, Identifikationscode bezeichnet. Damit man diesen Code von einem normalen Dateinamen unterscheiden kann, wird er normalerweise durch einen Doppelpunkt ":" abgeschlossen, wenn er in einer Basic-Anweisung eingegeben wird.

Diskettenlaufwerke: Ihnen werden Bezeichnungen im folgenden Format zugewiesen:

Dfn:

n ist eine einstellige Zahl, die die Nummer Ihres Laufwerkes angibt. Das erste Diskettenlaufwerk Ihres Systems (normalerweise das interne Laufwerk) bezeichnet man als Df0:, darauf folgen die Laufwerke Df1:, Df2: und Df3:, falls vorhanden.

Festplatte: Sie werden mit

Dhn:

bezeichnet. *n* stellt hier wieder die Nummer der Festplatte dar.

Volumes

Der Amiga erzeugt auch für jede Diskette eine eigene VOLUME-Bezeichnung. Sie können in jedem der AMOS Befehle die Laufwerksangabe auch durch diese Bezeichnung ersetzen. AMOS sucht dann alle verfügbaren Laufwerke nach der entsprechenden Diskette ab. Wenn sie nicht gefunden wird, erhalten Sie die Fehlermeldung *Device nicht ansprechbar.*

Wenn Sie eine neue Diskette von der Workbench aus vorbereiten, erhält diese Diskette stets die Bezeichnung "Empty". Diese Bezeichnung können Sie aus der Workbench heraus ganz leicht ändern. Klicken Sie einfach die Option RENAME an und geben Sie den neuen Namen über die vorhandene Dialogbox ein. Als Namen können Sie dabei praktisch jede beliebige Zeichenkette einsetzen, wenn Sie ihn jedoch in Ihren Programmen verwenden, muß er, genau wie der Laufwerksname, durch einen Doppelpunkt abgeschlossen werden. Hier sind ein paar typische Volume-Bezeichnungen für Sie:

AMOS:
AMOS_DATA:

Sie können zum Beispiel auf folgende Art eingesetzt werden:

Load "AMOS:Sprite.Acc" : Rem Lädt den Sprite-Editor von einer Diskette namens AMOS

Dir "AMOS_DATA:" : Rem Zeigt das Verzeichnis von AMOS_DATA: an

Achtung! Wenn Sie verschiedene Disketten mit der gleichen Bezeichnung versehen, oder die Disketten wild vertauschen, weiß der Amiga dann sehr bald nicht mehr, auf welche Diskette Sie sich beziehen. In diesem Fall müssen Sie stattdessen die Laufwerksbezeichnung eingeben. So erfährt AMOS ganz genau, wo sich die entsprechende Diskette in Ihrem System befindet. Dazu folgendes Beispiel:

Dir "Df0:"

Wir würden Ihnen stark dazu raten, wirklich jeder Diskette, mit der Sie arbeiten, eine eigene Bezeichnung zu geben. Das dauert in der Workbench nur ein paar Sekunden, wird Ihnen die Arbeit aber ungemein erleichtern.

Logische Einheiten

Und schließlich gibt es da noch eine Gruppe von Objekten, die man logische Einheiten nennt. Sie werden von den Betriebsroutinen des Amiga dazu verwendet, die genaue Position der wichtigen Systemdateien wie die Ein- und Ausgabeeinheiten und Fonts zu bestimmen. Jeder Einheit wird normalerweise auf der aktuellen Startdiskette ein bestimmtes Verzeichnis zugewiesen. Hier sind einige Beispiele, die von AMOS eingesetzt werden:

FONTS:	Ein Verzeichnis, das die aktuellen Fonts enthält
LIBS:	Enthält eine Library-File, auf die der AMOS SAY-Befehl zugreifen muß

Tippen Sie doch einmal die folgende Zeile aus dem Direktmodus ein:

Dir "LIBS:"

Cross Dos

Wenn Sie das eigenständige Cross-Dos-Paket gekauft und im Speicher installiert haben, so können Sie über AMOS-Basic auf Disketten im IBM- oder ST-Format zugreifen. Diesen Disketten werden Bezeichnungen zugewiesen, die mit den Buchstaben DI beginnen.

DIn:	(n stellt dabei die Nummer Ihres Laufwerkes dar)

Wenn Sie jetzt Ihre STOS-Programme zu AMOS-Basic konvertieren möchten, müssen Sie sie zuerst mit der Option FSAVE "*.ASC" in STOS im ASCII-Format speichern. Dann legen Sie die Diskette in ein Diskettenlaufwerk des Amiga ein, das über Cross Dos als IBM-Laufwerk eingerichtet wurde.

Achtung: Aufgrund der Unterschiede zwischen AMOS und STOS müssen Sie manche STOS-Programme erst etwas modifizieren, bevor sie mit AMOS laufen. Wenn Sie die Spezialfunktionen von AMOS-Basic einsetzen möchten, werden Sie bei gewissen Programmen sogar ganz durchgreifende Änderungen vornehmen müssen. Trotzdem lohnt sich die Mühe bei der Konvertierung Ihrer STOS- Programme in das AMOS-Format auf alle Fälle. Die verstärkte Leistung der Amiga- Hardware kann Ihre STOS-Spiele bis zur Unkenntlichkeit verbessern!

Ändern des Verzeichnisses

DIR *(Zeigt das Verzeichnis der aktuellen Diskette an)*

DIR [PFAD$]
DIR/W [PFAD$]

Durch diesen Befehl werden alle Dateien auf der aktuellen Diskette aufgeführt. Sie können wahlweise über den Parameter Pfad$ einen Pfad angeben, und nur die Dateien, die bestimmte Kriterien erfüllen, werden dann angezeigt. Alle Unterverzeichnisse in dem Listing werden durch ein "*" hervorgehoben.

Sie können das Listing jederzeit durch Drücken der Leertaste anhalten. Wenn Sie die Leertaste dann ein zweites Mal drücken, geht die Auflistung weiter.

Wenn Sie Disketten austauschen und versuchen, dann ein Verzeichnis abzurufen, erhalten Sie oft die Fehlermeldung *Device nicht ansprechbar*. Das kommt dadurch zustande, daß Sie die aktuelle Diskette gewechselt haben, ohne AMOS-Basic davon in Kenntnis zu setzen. Dieses Problem beseitigen Sie ganz einfach, indem Sie die Bezeichnung des Verzeichnisses mit dem Namen der neuen Diskette ändern. Das geschieht zum Beispiel, wenn Sie eingeben: DIR$="Df0:", bevor Sie DIR aufrufen.

Durch /W werden die Dateien in zwei Spalten auf dem Bildschirm dargestellt. Auf diese Art können doppelt soviele Dateien gleichzeitig angezeigt werden.

Die Zeichenkette für den Pfad besteht aus drei Hauptelementen:

[Disk:][Verzeichnis/] Filter

Disk Dies ist die Bezeichnung der Diskette, die untersucht werden soll. Damit Ihr Diskettenname nicht mit einem Dateinamen verwechselt werden kann, müssen Sie ihn mit einem Doppelpunkt abschließen. Hier ist ein Beispiel:

```
Dir "AMOS:"
Dir "Df0:"
Dir "FONTS:"
```

Verzeichnis/ Enthält die Bezeichnung eines Unterverzeichnisses, das Sie anzeigen möchten. Beispiele:

```
Dir "AMOS:IFF/"
Dir "IFF/"
```

Filter Definiert eine Reihe von Bedingungen, die von jeder Datei in Ihrem Listing erfüllt werden müssen.

Normaler Text: Jedes Zeichen in Ihrem Text sollte genau einem Zeichen in dem Dateinamen, der angezeigt werden soll, entsprechen. Dazu ein Beispiel:

Dir "Music"
Music

* (Sternchen): Vergleicht eine Reihe von Buchstaben in Ihren Dateinamen bis zum nächsten Steuerzeichen. Beispiel:

Dir "M*" : Rem Zeigt alle Dateien an, deren Namen mit M beginnen
Music
Mercury
Malt

Als Default ignoriert diese Option alle Dateien, die eine MS-DOS artige Dateierweiterung aufweisen. So wird zum Beispiel eine Datei mit der Bezeichnung Mad.Asc nicht aufgeführt.

. (Punkt): Vergleicht die Dateierweiterung. Das wird normalerweise in Verbindung mit dem "*"-Befehl eingesetzt, um alle Dateien mit einer bestimmten Erweiterung aufzuführen. Hier sind einige Beispiele dazu:

Dir "*.*" : Rem Führt alle Dateien auf der Diskette mit einer Erweiterung auf
Dir "*:Amos"" : Rem Listet alle AMOS-Dateien
Dir "Program.*" : Rem Zeigt alle Namen an, die mit Program beginnen

?: Vergleicht die Zeichen an der angegebenen Position. Also werden durch **Dir "AM??"** zum Beispiel folgende Dateien gelistet:

AMOS
AMAL

Eine Bezeichnung wie "AMOS-BASIC" wird jedoch ignoriert, denn sie ist mehr als vier Zeichen lang. Jeder Buchstabe im Dateinamen muß einem Zeichen in der Suchfolge entsprechen, damit die Datei angezeigt wird.

Siehe auch LDIR.

=DIR$= *(Ändert das aktuelle Verzeichnis)*

s$=DIR$
DIR$=s$

DIR$ enthält ein Verzeichnis, das als Ausgangspunkt für alle folgenden Disketten-Operationen, wie zum Beispiel Laden oder Speichern, dient. Es ist damit dem CD-Befehl im CLI sehr ähnlich, hat jedoch zusätzlich den Vorteil, daß Sie das Verzeichnis nicht nur ändern, sondern auch lesen können. Dazu folgendes Beispiel:

```
Print Dir$ : Rem Druckt aktuelles Verzeichnis aus
Dir$="AMOS:IFF/" : Rem Stellt das Unterverzeichnis IFF auf der Diskette AMOS ein
```

Wie bei der CD-Funktion, die diese Anweisung ersetzt, wird auch hier angenommen, daß alle Unterverzeichnisse zu demjenigen, in dem Sie gerade arbeiten, in Beziehung stehen. Angenommen Sie geben folgende Zeile ein:

```
Dir$="MANUAL/"
```

Damit wird das aktuelle Verzeichnis auf MANUAL gestellt. Wenn Sie dann versuchen, ein anderes Unterverzeichnis einzugeben, wie zum Beispiel FONTS über den Befehl DIR$=FONTS/", erhalten Sie die Fehlermeldung *Datei nicht gefunden.* Da FONTS nicht im Unterverzeichnis MANUAL gefunden werden kann, ergibt sich ein Fehler. Dieses Problem können Sie vermeiden, indem Sie die Bezeichnung der Diskette folgendermaßen in die Zuweisung miteinbeziehen:

```
Dir$="AMOS:FONTS/"
```

oder

```
Dir$="Df0:FONTS/"
```

=DISC INFO$ *(Gibt Kurzinformation über Diskette aus)*

=DISC INFO$("Name")

Bei dem Aufruf gibt DISC INFO eine Kurzinformation über die Diskette *Name* aus, bestehend aus dem Namen und der Anzahl der freien Bytes. Die Kurzinformation hat folgendes Format:

"Name der Disk:XXXXXXX"

Dabei ist XXXXXXX die Anzahl an freien Bytes auf dem Datenträger. Mittels des folgenden Kurzprogrammes können beide Informationen getrennt ausgelesen werden :

```
FLOPPY$=DISC INFO("DF0:")
C=Instr(A$,":")
NAME$=Left$(A$,C)
PLATZ=Val(A$,C+1)
Print FLOPPY$
Print "Name der Disk : ";NAME$;" / ";PLATZ;" Bytes frei."
```

PARENT *(Setzt den aktuellen Pfad ein Verzeichnis nach oben)*

Das Speichersystem des Amiga ermöglicht es Ihnen, die Verzeichnisse ineinander zu verschachteln. Auf diese Art können Sie Ihre Dateien sehr leicht nach einer Reihe von Kriterien anordnen. Sehen wir uns dazu ein kleines Beispiel an:

```
VERZEICHNISA/
  VERZEICHNISB/
      VERZEICHNISC/
  VERZEICHNISD/
```

Das vorstehende Diagramm stellt vier verschiedene Verzeichnisse (folder) auf der Diskette dar. VERZEICHNISA befindet sich im Hauptverzeichnis oder *root directory* und enthält VERZEICHNISB und VERZEICHNISD. In der Computersprache bezeichnet man deshalb VERZEICHNISA als das Mutterverzeichnis oder parent directory. Entsprechend ist VERZEICHNISB wiederum das *Parent Directory* für VERZEICHNISC.

Wie Sie sich vielleicht vorstellen können, kann man sich in diesen Verzeichnissen sehr leicht verirren. Wenn man dann PARENT aufruft, kann man automatisch Schritt für Schritt durch diese Verzeichnisse nach oben steigen, bis man wieder im Hauptverzeichnis oder Root Directory ankommt. Hier ist ein Beispiel dazu:

```
Dir$="/FF"
Dir
Parent
Dir
```

SET DIR *(Bestimmt den Stil, den DIR einsetzt)*

SET DIR n[,Filter$]

Hier bestimmen Sie die Art, auf die Ihr Verzeichnis gelistet wird. n ist dabei die Anzahl der Zeichen (von 1 - 100), die bei jedem Dateinamen angezeigt werden. Beachten Sie, daß dies **keine** Wirkung auf die tatsächliche Länge Ihrer Dateinamen hat. Dieser Parameter verändert nur die Art, wie sie auf dem Bildschirm dargestellt werden.

Filter ist eine Reihe von Pfadnamen, die bei Ihrer Suche **ausgenommen** werden sollen. Alle Dateinamen, die diesem Filter entsprechen, werden völlig ignoriert und nicht als Teil Ihres Verzeichnisses angezeigt. Damit können Sie zum Beispiel die störenden ".INFO" Dateien unterdrücken, die die von der Workbench eingesetzten Icondefinitionen enthalten.

Beachten Sie hier auch, daß Sie gut eine ganze Reihe von Dateipfaden gleichzeitig ignorieren können. Schließen Sie einfach den Dateinamen mit einem "/"-Zeichen ab. Als Default ist der Filter auf:

".INFO/*.INFO/*.*.INFO"

gesetzt. Dazu folgendes Beispiel:

```
Set Dir 5 : Rem Zeigt nur die ersten fünf Zeichen in Ihren Namen an
Dir
Set Dir 30,"" : Rem Kein Filter
Dir
```

Häufig verwendete Diskettenbefehle

=DFREE *(Freier Diskettenspeicher)*

f=DFREE

Weist den freien Speicherplatz auf der aktuellen Diskette in Bytes aus.

```
Print Dfree
```

MKDIR *(Erstellt ein Unterverzeichnis)*

MKDIR f$

Erstellt auf der Diskette ein neues Unterverzeichnis mit der Bezeichnung f$. Dazu folgendes Beispiel:

```
Mkdir "Df0:TEST"
Dir
```

KILL *(Löscht eine Datei auf der Diskette)*

KILL f$

Löscht die Datei f$ auf der aktuellen Diskette. **Achtung!** Was immer Sie mit diesem Befehl lösen, ist für immer weg!

RENAME *(Umbenennen einer Datei)*

RENAME alt$ TO neu$

Ändert den Namen einer Datei. Wenn bereits eine Datei mit dem neuen Namen existiert, so erhalten Sie eine Fehlermeldung.

Wie man eine Datei auswählt

=FSEL$ *(Wählt eine Datei aus)*

f$=FSEL$(Pfad$[,Default$][,Titel1$,Titel2$])

Mit der Funktion FSEL$ können Sie Ihre Dateien direkt von der Diskette auswählen. Dabei wird der normale AMOS Datei-Selektor eingesetzt.

Pfad$ gibt ein Suchmuster an, das bestimmt, welche Dateien in Ihrem Listing angezeigt werden.

Nachdem Sie eine Datei ausgewählt haben, zeigt FSEL$ entweder den komplette Pfad an, oder eine leere Zeichenkette "", wenn Sie QUIT gewählt haben.

Default$ wählt den Dateinamen, der als Default eingesetzt werden soll. Diese Datei wird dann automatisch gewählt, wenn der Anwender die Arbeit durch Drücken von ENTER abbricht.

Titel1$* und *Titel2$ sind Textketten, die wahlweise eingegeben werden können. Der eingegebene Titel wird dann oben an Ihrem Datei-Selektor angezeigt. Hier ist ein Beispiel dazu:

```
F$=FSEL$("*.IFF","","Lädt eine IFF-Datei")
If F$="" The Edit : Rem Zurück zum Editor wenn keine Datei gewählt wurde
Load IFF F$,0 : Rem Lädt Datei
```

Wie man ein AMOS-Programm von einer Diskette ablaufen läßt

RUN *(Führt ein AMOS Basic-Programm aus)*

RUN [Datei$]

Es ist zwar ganz einfach, Ihre AMOS-Programme direkt aus dem Editor heraus ablaufen zu lassen, aber wir haben trotzdem noch einen eigenen RUN-Befehl eingebaut. Dieser Befehl kann - ohne die Eingabe des Parameters *Datei$* - nur vom Direktmodus aus eingegeben werden.

Sie können jedoch eine RUN-*Datei*-Anweisung auch in ein Basic-Programm einbauen. Auf diese Art können Sie eine Reihe von Programmen miteinander *verketten.* Aber wenn Sie ein Programm auf diese Art ablaufen lassen, vergessen Sie nicht, daß das vorhandene Programm aus dem Speicher gelöscht wird, und die Variablen verlorengehen. Alle Datenschirme, die erzeugt wurden, bleiben jedoch intakt, und so können zwischenzeitlich Ladeschirme angezeigt werden. Dazu folgendes Beispiel:

```
Print "Test wird durchgeführt"
Run "Hithere.AMOS" : Rem Auf der AMOS-Programmdiskette
Print "Diese Zeile wird nie ausgeführt"
```

Dieser Befehl ist ungemein nützlich, denn Sie können auf diese Art jedes AMOS- Spiel in eine Reihe von Stufen aufteilen, die einzeln von der Diskette geladen werden. Jede Stufe kann dabei als ganz eigenständiges Programm geschrieben werden. Somit liegt die einzige Beschränkung für den Umfang Ihrer Spiele beim verfügbaren Speicherplatz. Und Sie können mit diesem System absolute Riesenspiele schreiben.
Siehe hierzu auch PRUN.

Wie man herausfindet, ob eine Datei existiert

=EXIST *(Überprüft, ob eine bestimmte Datei existiert)*

Flag=EXIST(f$)

EXIST sucht in dem aktuellen Verzeichnis nach der Datei f$. Wenn sie vorhanden ist, dann wird der Wert -1 (richtig) angezeigt, andernfalls erscheint 0 (falsch). Mit dieser EXIST-Funktion kann man alles überprüfen, vom Vorhandensein einer einzigen Datei bis zu einer ganzen Diskette. Hier sind einige Beispiele dazu:

```
Print Exist("Das ist wirklich ein blöder Name HAHA")
Print Exist("AMOS:") : Rem Ist eine Diskette namens AMOS: verfügbar
Print Exist("Df1:") : Rem Ist ein zweites Laufwerk angeschlossen
```

Wie Sie sehen, können Sie bei EXIST auch totalen Schwachsinn eingeben, ohne eine Fehlermeldung zu erhalten. Vorausgesetzt, der Dateiname enthält mindestens ein Zeichen, macht EXIST einfach weiter. Leere Zeichenketten wie "" sollten jedoch getrennt abgefragt werden. Dazu folgendes Beispiel:

```
F$=Fsel("*.IFF","","Lädt eine IFF-Datei")
If F$="" Then Edit : Rem Zurück zum Editor wenn keine Datei gewählt wurde
If Exist(F$) Then Load Iff F$,0
```

=DIR FIRST$ *(Gibt die erste Datei in dem Verzeichnis an, die der Pfadbezeichnung entspricht)*

Datei$=DIR FIRST$(Pfad$)

Weist eine Zeichenkette aus, die den Namen und die Länge der ersten Datei auf der Diskette enthält, die mit dem aktuellen Suchpfad Pfad$ übereinstimmt. Wenn man diese Funktion aufruft, dann wird das gesamte Directory Listing in den Speicher geladen. Sie können jetzt den Namen der nächsten Datei in dem Verzeichnis durch Aufrufen der Funktion DIR NEXT$ abrufen.

```
Print Dir First$("*.*")
```

DIR NEXT$ *(Zeigt die nächste Datei an, die mit dem aktuellen Pfad übereinstimmt)*

Datei$=DIR NEXT$

Gibt den nächsten Dateinamen in einem Verzeichnis an, das zuvor mit der Anweisung DIR FIRST$ abgefragt wurde. Nachdem die letzte Datei auf dieser Liste gelesen wurde, erscheint eine Zeichenkette, die eine leere Kette "" enthält. Nun wird das gesamte Array dieses Verzeichnisses gelöscht, und der Speicher, den es benötigte, wieder für den Rest Ihres Basic-Programmes verfügbar. Hier ist ein Beispiel, in dem alle Dateien im aktuellen Verzeichnis angezeigt werden:

```
F$=Dir First$("*:*") : Rem Liest Verzeichnis und erfaßt erste Datei
While F$<>""
   Print F$ : Bell : Wait 30
   F$=Dir Next$ : Rem Sucht nächste Datei auf der Liste
Wend
```

In **BEISPIEL 20.1** im MANUAL-Unterzeichnis finden Sie eine weitere Demonstration dieser Befehle. Hier wird der Inhalt eines Unterverzeichnisses von einer Diskette auf eine andere kopiert.

Dateien auf Disketten

Eine Datei ist eigentlich nur eine Sammlung von Informationen, die an einer Stelle auf der Diskette angeordnet wurde. Jeder Datei wird ein eigener Name zugewiesen, der zwischen 1 und 255 Zeichen lang sein kann.

Bevor Sie jedoch mit einer dieser Dateien arbeiten können, müssen Sie sie erst mit den Anweisungen OPEN IN, OPEN OUT oder APPEND initialisieren. Wenn Sie eine Datei öffnen, weisen Sie ihr eine *Kanal*nummer zu, die zwischen eins und zehn liegt. Diese Nummer wird dann auch bei allen folgenden Diskettenoperationen eingesetzt, um die Datei, mit der Sie gerade arbeiten, zu identifizieren.

Der Commodore Amiga unterstützt zwei verschiedene Arten von Plattendateien, die Sequenzdateien und die Direktzugriffs- oder RA-Dateien.

Die Sequenzdateien

Die Dateien, die normalerweise auf dem Amiga eingesetzt werden, sind Sequenzdateien. Sie haben ihren Namen dadurch erhalten, daß man die in ihnen enthaltenen Informationen nur genau in der Reihenfolge lesen kann, in der sie ursprünglich erstellt wurden.

Das bedeutet eigentlich, daß Sie, wenn Sie nur einen kleinen Teil der Daten in der Mitte einer Sequenzdatei verändern möchten, die ganze Datei bis zu einschließlich diesem Wert einlesen, und dann die ganze Datei wieder zurück auf die Diskette schreiben müssen.

In AMOS-Basic erhalten Sie nur entweder zum Schreiben oder Lesen Zugang zu den Sequenzdateien, niemals für beides gleichzeitig.

```
Open Out 1,"file.seq"
Input "Wie heißt du";N$
Print #1,N$
Close 1
```

So wird eine Datei mit der Bezeichnung FILE.SEQ erzeugt, die Ihren Namen enthält. Wenn Sie diese Information wieder aus der Datei lesen möchten, müssen Sie die folgenden Zeilen eingeben:

```
Open In 1,"file.seq"
Input #1,N$
Print "Ich erinnere mich an dich. Du heißt";N$
Close 1
```

Diese beiden Programme führen drei verschiedene Operationen durch:

- Öffnen der Datei entweder durch OPEN IN, OPEN OUT oder APPEND.
- Zugang zur Datei durch INPUT# oder PRINT#.
- Schließen der Datei durch CLOSE. Achtung: wenn Sie diese Anweisung vergessen, gehen alle Änderungen, die Sie in der Datei vorgenommen haben, verloren!

Diese drei Schritte müssen jedesmal, wenn Sie auf eine Sequenzdatei zugreifen, in genau dieser Reihenfolge durchgeführt werden,.

OPEN OUT *(Öffnet eine Datei zur Ausgabe)*

OPEN OUT Kanal,n$

Öffnet eine Sequenzdatei zur Erstellung. Wenn diese Datei bereits existiert, dann wird sie gelöscht. *Kanal* ist eine Zahl zwischen 1 und 10 und dient dazu, Ihre neue Datei bei allen folgenden PRINT#-Befehlen zu identifizieren.

n ist der Name der Datei, die eröffnet werden soll.

APPEND *(Fügt an das Ende einer vorhandenen Datei weitere Informationen hinzu)*

APPEND Kanal,Name$

APPEND öffnet eine Sequenzdatei für die Ausgabe. Wenn diese Datei bereits existiert, wird die Information am Ende angefügt. So können Sie Ihre Dateien jederzeit erweitern, nachdem Sie sie einmal definiert haben.

OPEN IN *(Öffnet eine Datei für die Ausgabe)*

OPEN IN Kanal,Name$

OPEN IN stellt eine Datei für das Lesen bereit. Wenn diese Datei noch nicht existiert, wird sie automatisch erzeugt.

Kanal ist wiederum eine Zahl zwischen 1 und 10, die von den verschiedenen INPUT-Anweisungen eingesetzt wird, um Ihre geöffnete Datei zu bezeichnen.

CLOSE *(Schließt eine Datei)*

CLOSE n

Schließt die Datei mit der Nummer *n*. **Achtung!** Wenn Sie vergessen, eine Datei zu schließen, nachdem Sie sie fertigbearbeitet haben, gehen alle Veränderungen, die Sie vorgenommen haben, vollständig verloren.

PRINT# *(Überträgt eine Reihe von Variablen auf eine Datei oder zu einer Ausgabeeinheit)*

PRINT# Kanal,Variablenliste

Dieser Befehl ist identisch zur normalen Print-Anweisung, aber anstatt die Information auf dem Bildschirm anzuzeigen, überträgt er sie auf eine Datei oder zu einer Ausgabeeinheit, die durch die Kanalnummer angegeben wird. Dazu folgendes Beispiel:

```
Open Out 1,"Testdatei"
Print#1,"Hallo"
Close 1
```

Genau wie PRINT können Sie auch PRINT# mit ?# abkürzen.

INPUT# *(Eingabe einer Reihe von Variablen aus einer Datei oder über eine Ausgabeeinheit)*

INPUT #Kanal,Variablenliste

INPUT# erfaßt Information entweder aus einer Sequenzdatei oder über eine Ausgabeeinheit wie zum Beispiel die seriellen Schnittstelle. Wie beim normalen INPUT-Befehl wird hier eine Reihe von Werten eingegeben und in einen Satz von Basic-Variablen geladen. Wie immer müssen dabei alle Werte durch Kommas voneinander abgetrennt werden. Dazu kommt, daß jede Zeile von Daten durch ein Zeichen für die Zeilenschaltung <line feed> abgeschlossen werden muß. Das entspricht dem ENTER, mit dem Sie eine Zeile abschließen, die sie über die Tastatur eingegeben haben. Hier ist ein Beispiel:

```
Open In 1,"Testdatei" : Rem Öffnet die im vorhergehenden Beispiel erstellte Datei
Input #1,A$
Print A$
Close 1
```

LINE INPUT

(Eingabe einer Reihe von Variablen, die nicht durch ein "," voneinander abgetrennt sind)

LINE INPUT # kann über zwei mögliche Formate verfügen:

LINE INPUT #Kanal, Variablenliste

oder

LINE INPUT #Kanal,Trennungszeichen,Variablenliste.

Diese Funktion ist identisch mit INPUT #, abgesehen davon, daß Sie hier Ihre Reihe von Daten mit jedem beliebigen Zeichen anstelle eines Kommas unterteilen können. Wenn keine Trennungszeichen eingegeben werden, so wird dafür automatisch das ENTER-Zeichen gesetzt.

Wenn Sie einen Text lesen, dann ist der Befehl LINE INPUT # immer vorzuziehen, denn die Kommas, die in dem Text als normale Satzzeichen dienen, werden vom INPUT#-Befehl als Trennungszeichen aufgefaßt. Und Ihr Programm ist dann davon völlig verwirrt.

SET INPUT

(Bestimmt das Zeichen für das Zeilenende)

SET INPUT c1,c2

Diese Anweisung setzt das Zeichen für das Zeilenende, durch das eine Datenzeile abgeschlossen wird. Der Amiga erwartet ein einziges <line feed> Zeichen am Ende jeder Zeile, während die meisten anderen Computer (einschließlich des ST) sowohl ein ENTER wie auch ein <line feed> benötigen. Wenn Sie also versuchen, Ihre ST-Dateien über ein serielles Kabel zu importieren, so erhalten Sie unzählige überflüssige ENTER-Zeichen in Ihren Dateien. Dieses Problem können Sie glücklicherweise umgehen, indem Sie den Befehl SET INPUT verwenden.

c1 und *c2* enthalten ein Paar ASCII-Werte, die als Trennungszeichen eingesetzt werden. Wenn Sie dafür ein einziges Zeichen verwenden möchten, so laden Sie einfach c2 mit einem negativen Wert wie zum Beispiel -1. Hier sind einige Beispiele dazu:

```
Set Input 10,-1 : Rem Normales Amiga-Format
Set Input 13,10 : Rem ST-Format
```

=INPUT$ *(Gibt eine Reihe von Zeichen über eine Ausgabeeinheit ein)*

x$=INPUT$(f,Anzahl)

Liest die *Anzahl* Zeichen der Ausgabeeinheit oder Datei mit der Nummer *f.*

=EOF *(Abfrage nach dem Dateiende)*

Flag=EOF (Kanal)

EOF ist eine nützliche AMOS Basic-Funktion, die überprüft, ob sich das Ende einer Datei an der aktuellen Position befindet. Wenn das Ende erreicht ist, weist EOF den Wert -1 aus, andernfalls wird 0 angezeigt.

LOF *(Länge der geöffneten Datei)*

Länge=LOF(Kanal)

Gibt die Länge der geöffneten Datei an. Es ist aber nicht sinnvoll, diese Funktion in Verbindung mit anderen Einheiten als Disketten anzuwenden.

POF *(Variable, die aktuelle Position des Dateizeigers enthält)*

Pos=POF(Kanal)

Mit der POF-Funktion kann man die aktuelle Position für das Lesen oder Schreiben in einer Datei verändern. Dazu ein Beispiel:

```
Pof(1)=1000
```

Hiermit wird die Schreib-/Leseposition auf 1000 Zeichen nach dem Dateianfang gesetzt. Eigenartigerweise kann man so über POF eine Art direkten Zugriff (Random Access) auf Sequenzdateien erhalten! Das funktioniert eigentlich nur deshalb, weil alle Laufwerke im Grunde direkt arbeiten und alle sequentiellen Operationen im Prinzip nur emuliert werden.

Direktzugriffsdateien (Random Access)

Wie der Name schon sagt, können Sie bei Direktzugriffs- oder RA-Dateien auf die auf der Diskette enthaltenen Information in jeder beliebigen Reihenfolge zugreifen. Bevor Sie diese Dateien jedoch einsetzen können, sollten Sie sich ein bißchen mit der Theorie vertraut machen.

Alle RA-Dateien setzen sich aus Einheiten zusammen, die man Datensätze nennt.

Jeder dieser Datensätze verfügt über eine eigene Nummer. Diese Sätze werden wiederum in eine Reihe von Feldern unterteilt. Und jedes Feld enthält nun eine bestimmte Information. Wenn Sie mit Sequenzdateien arbeiten, dann können diese Felder jede beliebige Länge haben, denn die Datei kann ja nur in eine Richtung gelesen werden. Bei RA-Dateien müssen Sie jedoch die maximale Größe jedes dieser Felder im Voraus bestimmen.

Angenommen, Sie möchten eine Datei erzeugen, die eine Reihe von Namen und Telefonnummern enthält. In diesem Fall könnten Sie die folgenden Felder einsetzen:

Feld	Maximale Länge
NACHNAMW$	15
NAME$	15
VORWAHL$	10
TEL$	10

Sie könnten diese Felder jetzt zum Beispiel mit folgender Zeile definieren:

```
Field #1,15 as NACHNAME$,15 as NAME$,10 as VORWAHL$,10 as TEL$
```

Dabei sollten Sie nicht vergessen, daß die Zeichenketten, die durch die FIELD-Anweisung definiert werden, auch als normale Zeichenkettenvariablen eingesetzt werden können. So können Sie bei jedem beliebigen Feld Informationen ein- und ausgeben. Dazu folgendes Beispiel:

```
NACHNAME$="BERGER" : Rem Lädt den Nachnamen in das Feld NACHNAME$.
TEST$=NACHNAME$ : Print TEST$
```

Nachdem Sie die gewünschten Informationen in Ihren Datensatz geladen haben, können Sie ihn mit dem PUT-Befehl auf die Diskette schreiben. Zum Beispiel so:

```
Put 1,10
```

Lädt die Daten in den Datensatz Nummer 10 der Datei, die im Kanal 1 geöffnet ist.

Entsprechend können Sie durch die GET-Anweisung einen Datensatz lesen.

```
Get 1,10
```

OPEN RANDOM *(Öffnet einen Kanal für eine Direktzugriffsdatei)*

OPEN RANDOM Kanal,n$

Öffnet eine Direktzugriffsdatei mit der Bezeichnung *n$* auf der aktuellen Diskette. Wenn

Sie diese Anweisung einsetzen, sollten Sie die Struktur des Datensatzes immer sofort danach mit dem FIELD$-Befehl definieren.

FIELD *(Definiert die Struktur des Datensatzes)*

FIELD Kanal, Länge1 AS Feld1$, Länge2 AS Feld2$...

Über FIELD können Sie einen Datensatz definieren, der dann für eine Direktzugriffsdatei eingesetzt wird. Dieser Datensatz kann bis zu 65535 Bytes lang sein.

```
Field 1,15 as NACHNAME$,15 as NAME$,10 as VORWAHL$,10 as TEL$
```

PUT *(Ausgabe eines Datensatzes in eine Direktzugriffsdatei)*

PUT Kanal,r

PUT bewegt einen Datensatz vom Speicher des Amiga in den Datensatz mit der Nummer *r* einer RA-Datei. Bevor Sie jedoch mit dem Inhalt des neuen Datensatzes arbeiten, sollten Sie ihn zuerst in die mit FIELD definierten Zeichenketten für die Felder stellen. Dabei beginnen Sie zum Beispiel mit einer solchen Anweisung:

```
NACHNAME$="BERGER"
```

Sie können die vorhandenen Datensätze zwar in jeder beliebigen Reihenfolge eintragen, aber Sie dürfen die Datensätze nicht völlig willkürlich auf der Diskette verstreuen. Das bedeutet, wenn Sie zum Beispiel gerade eine Datei erstellt haben, dürfen Sie zum Beispiel nicht so etwas eintippen:

```
Put 1,1
Put 1,5
```

In diesem Fall ruft die Anweisung Put 1,5 eine Fehlermeldung hervor, weil sich in der Datei noch keine Datensätze mit den Nummern 2 bis 5 befinden. Der Satz 2 muß also der nächste neue Satz in der Datei sein, dann folgen 3, 4 usw. Dazu folgendes Beispiel:

```
Open Random 1,"TELEFON"
Field 1,30 As NAME$,30 As TEL$
INDEX=1
Do
  Input "Namen eingeben";NAME$
  Exit If NAME$=""
  Input "Telefonnummer eingeben";TEL$
  Put 1,Index : Inc INDEX
Loop
Close 1
```

GET *(Eingabe eines Datensatzes aus einer RA-Datei)*

GET Kanal,r

GET liest den Datensatz mit der Nummer *r*, der sich in einer durch OPEN geöffneten RA-Datei befindet. Danach wird dieser Datensatz in die Zeichenketten der durch FIELD erzeugten Felder geladen. Diese Zeichenketten können jetzt wie gewohnt weiterbearbeitet werden.

Sie können mit GET jedoch nur Datensätze laden, die sich tatsächlich auf der Diskette befinden. Wenn Sie versuchen, einen Datensatz zu holen, der nicht existiert, erhalten Sie eine Fehlermeldung. Dazu folgendes Beispiel:

```
Open Random 1,"TELEFON"
Field 1,30 As NAME$, 30 As TEL$
Do
   Input "Datenzsatznummer eingeben";INDEX
   Exit If INDEX=0
   Get 1,INDEX : Print NAME$ :Print TEL$
Loop
Close 1
```

Der Drucker

Wenn Sie einen Drucker haben, können Sie die vollständigen Listings all Ihrer AMOS-Programme ausdrucken. Das ist eine ungemein wertvolle Hilfe, wenn Sie die Bugs beseitigen wollen und alle ernsthaften Programmierer sollten sich unbedingt einen Drucker beschaffen. Die Steuerung des Druckers geht über die folgenden Befehle ganz leicht:

LPRINT *(Gibt eine Reihe von Variablen über den Drucker aus)*

LPRINT Variablenliste

LPRINT ist im Prinzip die gleiche Anweisung wie PRINT, abgesehen davon, daß die Ausgabe hier nicht auf dem Bildschirm, sondern über den Drucker erfolgt. Dazu folgendes Beispiel:

```
Lprint "Hallo"
```

Siehe auch die Befehle PRINT, USING und PRINT#.

LDIR *(Schickt ein Verzeichnis an den Drucker)*

LDIR [PFAD$] [/W]
Das Verzeichnis der aktuellen Diskette wird über den Drucker ausgegeben. Weitere Einzelheiten siehe unter DIR.

Externe Ausgabeeinheiten

OPEN PORT *(Öffnet einen Kanal zu einer I/O-Schnittstelle)*

OPEN PORT Kanal,"PAR:" (Öffnet Kanal zur Parallelschnittstelle)
OPEN PORT Kanal,"SER:" (Öffnet Kanal zur seriellen Schnittstelle)
OPEN PORT Kanal,"PRT:" (Öffnet Kanal für den ausgewählten Drucker)

Über OPEN PORT können Sie mit externen Ausgabeeinheiten wie der RS232-Schnittstelle kommunizieren. Alle Standardbefehle für die Sequenzdateien können wie gewöhnlich eingesetzt werden, abgesehen von Befehle wie LOF oder POF, die natürlich nur für Diskettenoperationen zutreffen. Ein Beispiel:

```
Open Port 1,"SER:"
ForX=0 To 10
   Print #1,"AMOS-BASIC"
Next X
Close 1
```

Dieses Programm druckt zehn Textzeilen über die an die RS232-Schnittstelle angeschlossene Ausgabeeinheit aus. Wenn Ihr Printer über die Parallelschnittstelle läuft, verändern Sie die Zeile 10 folgendermaßen:

```
Open Port 1,"PRT:"
```

Entsprechend können Sie auch Informationen zum Beispiel über ein Modem folgendermaßen empfangen:

```
Input #1,A$ : Print A$
```

Wenn Sie mit diesen externen Ausgabeeinheiten arbeiten, können Sie alle normalen Input-Anweisungen einsetzen, einschließlich INPUT$ und LINE INPUT.

=PORT *(Überprüft, ob ein Kanal bereit ist)*

n=PORT(Kanal)

Überprüft, ob eine Eingabeeinheit zur Übertragung von Informationen bereit ist. Wenn diese Einheit auf die Abfrage wartet, so zeigt diese Funktion den Wert -1 an, andernfalls erscheint 0.

Das Suchen der aktuellen Devices

Sie können die aktuelle Device-Liste über die folgenden AMOS-Befehle abrufen:

=DEV FIRST$ *(Holt das erste Device aus der aktuellen Device-Liste)*

dev$=DEV FIRST$("filter")

Funktioniert im Prinzip genauso wie Dir First$ und Dir Next$, weist jedoch die Device-Liste aus. Beachten Sie, daß die Leerschritte durch -" " entfernt werden sollten, damit Sie den richtigen Namen erhalten.

=DEV NEXT$ *(Erfaßt das nächste Device, das dem Filter entspricht)*

dev$=DEV NEXT$

Erfaßt das nächste Device auf der Device-Liste. Eine Null-Zeichenkette zeigt an, daß das Ende der Liste erreicht ist. Zum Beispiel:

```
Print Dev First$
Do
  A$=Dev Next$
  A$=A$-" `
  If A$=''''then End
  Print A$
Loop
```

21: Gepackte Bildschirme

Die Amiga-Hardware kann zwar ganz phantastische Bilder darstellen, aber es war bisher so gut wie unmöglich, diese Effekte in einem richtigen Spiel zu erzielen. Das Problem dabei war in erster Linie der Speicher. Ein einziger 320x255 64- Farbenschirm frißt allein schon ganze 60 KByte RAM-Speicher. Damit ist die Anzahl der Schirme, die Sie in Ihren Spielen einsetzen können, stark eingeschränkt, vor allem, wenn die Programme dann auf einem A500 ohne Speichererweiterung laufen sollen.

Eine Lösung besteht darin, daß Sie Ihre Schirme mit dem AMOS Map-Definer-Zusatz generieren. Das ist allerdings nur für die Hintergrundschirme in einem Ballerspiel ganz annehmbar; für die realistischen Bilder, die Sie in einem Abenteuerspiel erwarten, eignet es sich nicht so gut.

Glücklicherweise stellen hier die AMOS Pack-Anweisungen eine einfache Alternative dar. Auf diese Art können Sie einen ganzen Schirm schnell zu einem Bruchteil seiner ursprünglichen Größe komprimieren. Alle Standard-Grafikmodi werden unterstützt, einschließlich HAM. Also können Sie ganz leicht irrsinnig starke Bilder in Ihre AMOS-Programme einbauen.

SPACK *(Komprimiert den Schirm)*

SPACK s TO n [tx,ty,bx,by]

Der Befehl SPACK (wird übrigens S-Pack ausgesprochen) packt den Schirm *s* in die Speicherbank *n*. Alles, was zu dem aktuellen Bild gehört, wird gespeichert, einschließlich von Modus, Schirmgröße, Offset und Anzeigeposition. Auf diese Art können Sie dann Ihren Schirm wieder genau in seinem Originalzustand zurückversetzen.

s ist die Nummer des Bildschirms, der Ihr Bild enthält. *n* stellt die Nummer einer der Speicherbanken 1 bis 16 dar. Wenn diese Bank noch nicht bereits existiert, so wird sie nun automatisch für Sie reserviert. Wenn möglich, wird für diese neue Bank FAST-Speicher eingesetzt. Sie wird zusammen mit Ihrem Basic-Programm gespeichert. Nachdem Sie diese Funktion aufgerufen haben, können Sie die Größe Ihres Schirms mit LENGTH abfragen. Dazu folgendes Beispiel:

```
F$=FSEL$("*","","Lädt ein Bild")
Load Iff F$,0
Spack 0 To 1 : Rem Komprimiert Schirm Null in eine Datei in Bank eins
Print "Die Länge der neuen Bank beträgt";Length(1);" Bytes""
Wait Key
Screen Close 0
Unpack 1 To 0 : Rem Entpackt gepackten Schirm wieder
```

Sie müssen mit dieser Anweisung natürlich nicht den ganzen Schirm packen. Durch die Parameter können Sie auch wahlweise jeden beliebigen rechteckigen Bildschirmabschnitt bestimmen

tx,ty stellen jetzt die Koordinaten der linken oberen Ecke dieses Bereichs dar. *bx,by* bestimmen die Position der rechten unteren Ecke Ihres Bildes. Alle X-Koordinaten werden dabei auf die nächstliegende 16-Pixel-Grenze gerundet.

Um den erforderlichen Speicherplatz so gering wie möglich zu halten, versucht SPACK erst einmal, Ihr Bild auf mehrere verschiedene Arten zu packen. Dann wird Ihr Bild schließlich mit der Methode gepackt, die am wenigsten Speicher benötigt. Eine Nebenwirkung dieser gründlichen Vorgehensweise besteht jedoch darin, daß es ungefähr 6 Sekunden dauert, eines Ihrer Bilder zu packen. Das ist aber eigentlich kein so großer Nachteil, weil Sie das Packen normalerweise nur einsetzen, während Sie Ihre Programme schreiben.

Da jedes Bild auf dem Schirm in weniger als einer Sekunde wieder entpackt werden kann, besteht keine Gefahr, daß die Geschwindigkeit Ihrer Spiele in Mitleidenschaft gezogen wird. Wenn die Geschwindigkeit jedoch ein besonderer Knackpunkt ist, dann wollen Sie vielleicht lieber das CBLOCK-System einsetzen. In dem Kapitel über Hintergrundgrafik finden Sie weitere Einzelheiten dazu.

Wenn Sie übrigens die gepackte Größe der Dateien mit ihrer ursprünglichen Länge auf der Diskette vergleichen, dann kann es sein, daß Sie den Umfang des gesparten Speichers unterschätzen. Sie dürfen dabei nämlich nicht vergessen, daß die meisten dieser Dateien durch die normalen IFF-Packroutinen BEREITS KOMPRIMIERT WURDEN. Also ist es ziemlich erstaunlich, daß AMOS sie nun noch einmal um 20% reduzieren kann!

Komprimierte Schirme eignen sich besonders gut für die Titel und Punktstandtabellen, die Sie für ein Ballerspiel brauchen. Sie können hier nämlich irre Schirmeffekte einbauen, ohne dabei zu viel Speicher zu verbrauchen. Außerdem können Sie die Schirme dann direkt in Rollen- und Abenteuerspiele einbauen.

PACK *(Packt einen Schirm)*

PACK s TO n [tx,ty,bx,by]

PACK komprimiert den Schirm *s* in die Speicherbank mit der Nummer *n*. Im Unterschied zum vorher besprochenen SPACK-Befehl werden hier nur die Bilddaten gepackt. Deshalb muß Ihr gepackter Schirm stets direkt in einen bereits vorhandenen Schirm entpackt werden.

Durch die Art, wie die Bilder entpackt werden, entsteht ein merklicher Flimmereffekt, wenn Sie Ihre Schirme nicht mit Double-Buffering gepuffert haben. Deshalb ist es ratsam, PACK nicht bei Schirmen mit Single-Buffering einzusetzen. In diesem Fall ist der SPACK-Befehl vorzuziehen.

Wenn Sie die Koordinaten eingeben, dann wird nur der so bezeichnete Bildschirmabschnitt gepackt. *tx,ty* geben die Position der linken oberen Ecke dieses Bereichs an und *bx,by* stellen die Koordinaten der rechten unteren Ecke Ihres Bildes dar. Wie zuvor, werden auch hier alle x-Koordinaten auf die nächstliegende 8-Pixel- Grenze gerundet.

Da PACK zu dem normalen Autoback-System völlig kompatibel ist, kann man gepackte Bilder leicht miteinander kombinieren, ohne die Schirme zu bewegen. Wenn Sie im zweiten Autoback-Modus arbeiten, können Sie Ihre Bilder sogar HINTER den vorhandenen BOBs entpacken. Sie können daher auch diese Anweisung in Verbindung mit SCREEN OFFSET einsetzen, um einen phantastischen Scroll-Hintergrund für Ihre Spiele zu erzeugen.

UNPACK *Entpackt einen gepackten Schirm)*

Durch diese Anweisung wird ein Schirm, der zuvor mit der SPACK- oder PACK-Anweisung gepackt wurde, wieder entpackt. Jede Packroutine verfügt über ihre eigene Ausführung des UNPACK-Befehls.

SPACK

UNPACK b TO s

Öffnet den Schirm s und bringt den gepackten Schirm wieder in die Bank *b* zurück. Wenn dieser Schirm bereits existiert, wird er von dem neuen Bild völlig ersetzt. Nachdem der Schirm dann entpackt wurde, wird er glatt ins Blickfeld geschoben.

PACK

GePACKte Schirm können durch zwei verschiedene Anweisungen wieder entpackt werden. Diese Anweisungen entnehmen einer Speicherbank ein Bild und laden es direkt in einen vorhandenen Schirm. **Achtung!** Der Zielschirm **muß** über genau dasselbe Format verfügen wie Ihr gepacktes Bild, sonst erhalten Sie die Fehlermeldung *Falscher Funktionsaufruf.*

Wenn Sie Ihre Schirme entpacken, fällt Ihnen vielleicht auf, daß die Sache ein bißchen unsauber aussieht. Das liegt daran, daß der PACK-Befehl eigentlich nur in Verbindung mit dem Double-Buffering-System eingesetzt werden soll. Wenn Sie Double-Buffering für Ihren Schirm verwendet haben, dann geht das Entpacken ganz wunderbar glatt über die Bühne.

UNPACK b

Entpackt den Schirm an seiner ursprünglichen Position.

UNPACK b,x,y

Zeichnet Ihr Bild neu und fängt dabei bei den Koordinaten *x,y* an. Wenn das neue Bild nicht in den aktuellen Schirm paßt, erhalten Sie die entsprechende Fehlermeldung.



22: Die Maschinensprache

AMOS-Basic enthält eine Reihe von Befehlen, die dem fortgeschrittenen Programmierer totalen Zugang zum Innenleben der Amiga-Hardware bieten. Im Allgemeinen brauchen Sie keinen dieser Befehle wirklich zum Programmieren.

Wir haben AMOS-Basic bewußt so gestaltet, daß Sie alle Eigenschaften des Amiga mit einfachen, leicht verständlichen Basic-Anweisungen ausnützen können. Deshalb brauchen Sie also im Prinzip nicht im Speicher herumzuhacken, um Ihre Spiele zu schreiben!

Die Umwandlung von Zahlen

=HEX$ *(Wandelt eine Zahl in einen Hexadezimalwert um)*

h$=HEX$(v)
h$=HEX$(v,n)

HEX$ wandelt die Ganzzahl *v* in einen Hexadezimalwert (Basis 16) um. Durch diese Anweisung wird eine Folge von *n* Hexadezimalzeichen in der Zeichenkette *h$* angezeigt. Hier folgen einige Beispiele dazu:

```
Print Hex$(Colour(1),3)
$A40
Print Hex$(65536)
$10000
Print Hex$(65536,8)
$00010000
```

=BIN$ *(Konvertiert eine Zahl in eine Binärkette)*

b$=BIN$(v)
b$=BIN$(v,n)

Wandelt eine Zahl in einen Binärwert (Basis 2) um. Wie bei HEX$ können Sie auch hier wählen, ob alle Stellen dieser Zahl angegeben werden sollen, oder nur einige. Ein Beispiel:

```
Print Bin$(255)
%11111111
Print Bin$(255,16)
%0000000011111111
```

n stellt dabei die Anzahl der Binärstellen dar, die in *b* angezeigt werden. Jeder Wert kann

1 bis 32 Stellen umfassen (1<=n<=31).

Speichermanipulation

=PEEK *(Erfaßt Byte an bestimmter Adresse)*

v=PEEK(Adresse)

Weist das an der Adresse gespeicherte 8-BitByte aus.

```
Load "AMOS_DATA:Sprites/Monkey.abk"
Print Peek(Start(1))
For S=8 To 1 Step -1
   C=peek(start(1)-S) : Print CHR$(c) ; : Rem Also da ist der Name versteckt
Next S
```

POKE *(Verändert Byte an bestimmter Adresse)*

POKE Adresse,v

Kopiert die Zahl *v* in die Adresse. *v* muß dabei stets im Bereich zwischen 0 und 255 liegen.

Mit dieser Funktion können Sie den Inhalt eines beliebigen Teil des Amiga-Speichers verändern. Aber vergessen Sie dabei nicht, daß POKE sehr gefährlich sein kann. Wenn Sie einfach ganz ungezielt herumpoken, dann bringen Sie den Amiga damit sehr schnell zum Absturz. Hier ist ein Beispiel:

```
Load "AMOS_DATA:Sprites/Octopus.abk"
Poke (Start(1)-5),ASC("o") : Rem Verändert einen Buchstaben im Namen der Bank
Listbank
```

=DEEK *(Erfaßt Wort an bestimmter Adresse)*

v=DEEK(Adresse)

Liest das zwei Bytes lange Wort an der angegebenen *Adresse*. Dabei muß *Addresse* ein gerader Wert sein, sonst erhalten Sie eine entsprechende Fehlermeldung.

DOKE *(Verändert Wort an der Adresse)*

DOKE Adresse,Wert

DOKE lädt einen 2 Bytes umfassenden *Wert* zwischen 0 und 65535 in die durch den Parameter *Addresse* vorgegebene Speicheradresse. In kundigen Händen ist diese Funktion äußerst nützlich. Da aber selbst die Besten unter uns manchmal Fehler machen, sollten Sie immer eine Kopie Ihres Programmes auf Diskette speichern, bevor Sie versuchen, diese Funktion in einer neuen Routine einzusetzen. Dazu folgendes Beispiel:

```
Doke Phybase(1)+1000,65535
```

=LEEK *(Liest ein langes Wort an der vorgegebenen Adresse)*

v=LEEK(Adresse)

Gibt das vier Bytes lange Wort an, das bei der angegebenen *Adresse* gespeichert ist. Wie bei DEEK, muß auch hier die betreffende Adresse GERADE sein. Wenn das Bit 31 des Ausgabewertes auf 1 gestellt wird, so wird *v* als negativer Wert dargestellt. Dabei handelt es sich nicht um einen Bug. Das ist nur eine Nebenwirkung der Art, wie AMOS mit Zahlen arbeitet. Ein Beispiel:

```
Print Leek(0)
```

LOKE *(Verändert langes Wort an der Adresse)*

LOKE Adresse,n

LOKE kopiert den vier Bytes umfassenden Wert *n* zu der Adresse. Beispiel:

```
Loke Phybase(1)+10000,$FFFFFFFF
```

Wenn Sie diese Funktion nicht völlig gezielt einsetzen, dann stürzt Ihr Amiga ganz furchtbar ab. Also Vorsicht!

=VARPTR *(Gibt Adresse einer Variablen an)*

Adresse=VARPTR(Variable)

Gibt die Speicheradresse eine Basic-Variable an. Dabei wird jeder Variablentyp in einem eigenen Format gespeichert.

Integer-Variablen: VARPTR ermittelt die Adresse der vier Bytes, die den Inhalt Ihrer Variablen enthalten. Dazu folgendes Beispiel:

```
A=0
Loke Varptr(A),1000
Print A
1000
```

Fließkomma-Variablen: VARPTR ermittelt die Adresse der vier Bytes, die den Wert Ihrer Variablen enthalten im IEEE-Format mit einfacher Genauigkeit an.
Zeichenketten-Variablen: Die VARPTR-Adresse weist auf das erste Zeichen in der Kette. Da AMOS-Basic die Zeichenketten nicht mit einem CHR$(0) abschließt, müssen Sie die Länge der Zeichenkette zum Beispiel mit DEEK(VARPTR(A$)-2) abfragen, dabei stellt A$ den Namen Ihrer Variable dar. Sie können natürlich auch LEN(A$) einsetzen.

COPY *(Kopiert einen Speicherblock)*

COPY Start,Ende TO Ziel

Mit diesem Befehl können Sie schnell große Abschnitte des Amiga-Speichers von einer Stelle an eine andere versetzen. *Start* und Ende sind dabei jeweils die Adressen der ersten und letzten Bytes Ihrer Daten. *Ziel* weist auf den Speicherbereich, in den Ihre neuen Daten geladen werden.

Alle diese Adressen **müssen** gerade sein, sonst erhalten Sie eine Fehlermeldung. Beispiel:

```
Reserve As Data 10,1000
A$="Hallo du"
Copy Varptr(A$),Varptr(A$)+Len(A$) To Start(10)
For P=Ø To Len(A$)
   Print Chr$(Peek(Start(10)+P));
Next P
```

FILL *(Füllt Speicherblock mit einem langen Wort)*

FILL Start TO Ende,Muster

Füllt einen bestimmten Speicherbereich mit den in Muster enthaltenen vier Bytes

Start und *Ende* bestimmen die Position und Größe des Blockes, der gefüllt werden soll. **Achtung!** Diese Adressen MÜSSEN gerade sein.

Muster ist ein langes Wort, das ein Füllmuster von vier Bytes enthält. Es wird dann in jede Gruppe von vier Speicherbereichen zwischen *Start* und *Ende* kopiert. Dazu folgendes Beispiel:

```
Reserve As Data 10,1000
Rem Erzeugt eine Kette aus vier Zeichen (4 Bytes = 1 langes Wort)
A$="AMOS"
Rem Füllt Bank mit Kette
Fill Start(10) to Start(10)+Length(10),Leek(Varptr(A$))
For P=0 To Length(10)
   Print Chr$(Peek(Start(10)+P));
Next P
```

=HUNT *(Findet eine Zeichenkette im Speicher)*

f=HUNT(Start TO Ende, S$)

Sucht den Amiga-Speicher nach der in *s$* enthaltenen Zeichenfolge ab. *Start* ist die Adresse des ersten Bytes, das im Speicher gesucht werden soll, und *Ende* die Adresse des letzten.

Nachdem die Anweisung ausgeführt wurde, enthält *f* entweder 0 (wenn die Zeichenkette in a$ nicht gefunden wurde) oder die Adresse von *f$*.

Bit-Operationen

ROL *(Rotieren nach links)*

ROL.B n,v
ROL.W n,v
ROL.L n,v

ROL ist die Basic-Version der ROL-Anweisung in der 68000 Assembler-Sprache. Sie hat folgende Wirkung: die Binärdarstellung eines Wertes in *v* wird erfaßt, und genau *n* Stellen nach links rotiert.

Wenn *v* nur aus einem einzigen Wert besteht, dann wird die Zahl, die rotiert werden soll, direkt aus *v* entnommen. Stellt *v* jedoch einen Ausdruck dar, so wird *v* stattdessen wie die **Adresse** Ihrer Zahl behandelt. Hier ist ein Beispiel dazu:

Die Zahl 136 wird folgendermaßen binär dargestellt:

%10001000

Tippen Sie jetzt ein:

```
V=136
Rol.B 1,V
Print Bin$(V,8)
%00010001
```

Beachten Sie, wie das erste Bit dieser Zahl wie durch Zauberei von hinten wieder nach vorne kam. Und nun wollen wir uns ein Beispiel für das Adressen-System ansehen:

```
Reserve As Work 10,1000
Poke Start(10),%00001111
Rol.B 1,Start(10)
Print Bin$(Peek(Start(10)),8)
%00011110
```

Dadurch erhalten Sie die Zahl 17, oder binär dargestellt, %00010001.

Wie Sie sehen, wurde die ganze Zahl nach links verschoben, wobei die höchste 1 in die niedrigste Position rotiert wurde. Das ".B" in der Anweisung teilt AMOS mit, daß diese Zahl als 8-Bit-Byte behandelt werden soll. Sie können den Umfang außerdem mit ".W" (Wort) und ".L" (langes Wort) angeben. Einige Beispiele dazu:

```
Rem Beachte den . zwischen Rol und L
A=1
Rol.L 1,A
Print A
2
```

```
Rem Beachte den . zwischen Rol und W
A=32768
Print BIN$(A)
Rol.W 2,A
Print A
2
```

Mit ROL können Sie schnell eine positive Zahl mit 2 multiplizieren.

ROR *(Nach rechts rotieren)*

ROR.B n,v
ROR.W n,v
ROR.L n,v

Dieser Befehl ist ROL sehr ähnlich, rotiert jedoch die Zahl in die entgegengesetzte Richtung. Wie zuvor kann v auch hier entweder eine einfache Variable oder einen Ausdruck darstellen. Wenn es ein Ausdruck ist, dann rotiert ROR die Zahl, die an der entsprechenden **Adresse** gespeichert ist. Beispiel:

```
A=8
Ror 1,A
Print A
4
```

ROR kann auch benutzt werden, um eine positive Zahl durch 2 zu dividieren. Die Berechnung wird dabei wesentlich schneller durchgeführt, als wenn sie mittels / aufgerufen würde.

=BTST *(Fragt ein Bit ab)*

b=BTST(n,v)

Fragt die binäre Stelle an der Position *n* der Variable *v* ab. Wenn v ein Ausdruck ist, wird er als Adresse des Bits, das abgefragt werden soll, eingesetzt. In diesem Fall wird *n* automatisch durch ein logisches UND mit 7 verknüpft, bevor weitergearbeitet wird.

Nachdem BTST aufgerufen wurde, wird *b* mit -1 (richtig) geladen, wenn das Bit an der Position *n* auf eins gestellt ist, andernfalls lautet das Ergebnis 0 (falsch). Beispiel:

```
B=%1010
Print Btst (3,B)
-1
Print Btst (2,B)
0
```

Siehe auch BCHG, BCLR und BSET.

BSET *(Stellt ein Bit auf 1)*

BSET n,v

Stellt das Bit an der Position *n* in der Variable *v* auf eins. Wenn Sie diese Variable durch einen Ausdruck ersetzen, dann wird sie wie die Speicheradresse eines Wertes behandelt. Dazu folgendes Beispiel:

```
A=0
Bset 8,A
Print A
256
```

BCHG *(Verändert ein Bit)*

BCHG n,v

Verändert das Bit mit der Nummer *n* in der Variable *v*. Wenn dieses Bit zur Zeit 1 beträgt, dann lautet der neue Wert Null, und umgekehrt. Wie immer wird das Ergebnis als Adresse betrachtet, wenn *v* einen Ausdruck enthält. Beispiel:

```
A=0
Bchg 1,A
Print A
2
```

BCLR *(Löscht ein Bit)*

BCLR n,v

Löscht das Bit mit der Nummer n in der Variable v, indem es auf Null gesetzt wird. Wie alle Bit-Operationen stellt v auch hier eine Speicheradresse dar, wenn es einen Ausdruck enthält. Hier ist ein Beispiel dazu:

```
A=128
Bclr 7,A
Print A
0
```

Das Arbeiten mit Assembler

AMOS-Basic enthält besondere Werkzeuge, durch die Sie Assembler-Routinen mit Ihren Basic-Programmen verbinden können. Dabei möchten wir allerdings betonen, daß Sie den Maschinencode eigentlich nur ganz selten wirklich brauchen, da das AMOS-System so leistungsstark ist. Im Grunde haben wir diese Funktionen nur für die Assembler-Programmierer eingebaut, die Ihre Basic-Programme manchmal durch ein bißchen Maschinensprache optimieren möchten.

Aber Vorsicht! Diese Befehle sind extrem gefährlich, und wenn Sie nicht vorsichtig mit ihnen umgehen, dann stürzt Ihr Amiga schnell ab. Wir würden Ihnen empfehlen, diese Funktionen zu vermeiden, wenn Sie mit den Eigenheiten der Amiga-Hardware nicht völlig vertraut sind.

PLOAD *(Reserviert eine Speicherbank für Maschinencode)*

PLOAD "Dateiname",Bank

Reserviert eine Speicherbank und lädt ein Programm in Maschinensprache von der Diskette in diese Bank.

Bank ist die Nummer der Speicherbank, die für Ihr Programm reserviert werden soll. Wenn dieser Wert negativ ist, dann wird der absolute Wert dieser Zahl für die Bank berechnet, und der erforderliche Speicherbereich aus dem CHIP-Speicher zur Verfügung gestellt

Nachdem Sie ein Programm einmal auf diese Art geladen haben, können Sie es als normale ".ABK"-Datei auf Diskette speichern. Da die so erzeugten Banken permanent sind, wird es auch direkt mit Ihren AMOS-Programmen gespeichert.

Ihr Programm muß aus einer Datei in Maschinensprache im normalen Amiga-Format bestehen. In der Praxis kann es so ziemlich alles mögliche enthalten, es bestehen jedoch die folgenden Beschränkungen:

- Der Code **muß** total beweglich (relocatable) sein, denn er wird im ersten Speicherbereich untergebracht, der verfügbar ist.
- Nur der **Code-**Teil Ihres Programmes wird geladen.
- Das Programm muß durch eine einzige RTS-Anweisung abgeschlossen werden.

CALL *(Ruft ein Programm in Maschinensprache auf)*

CALL Adresse[,Params]
CALL Bank[,Params]

Durch diese Anweisung wird ein Assembler-Programm, das sich im Amiga-Speicher befindet, ausgeführt.

Adresse kann entweder die absolute Adresse Ihres Codes darstellen, oder die Nummer einer AMOS Speicherbank, die zuvor mit PLOAD erzeugt wurde.

Am Anfang Ihres Programmes werden die Register D0 bis D7 und A0 bis A2 mit den Werten geladen, die in den DREG- und AREG-Arrays gespeichert sind. Jetzt kann Ihr Assembler-Programm jedes der 68000 Register beliebig verändern. Zu Beginn der Routine weist das Register A3 auf die Liste der wählbaren Parameter hin, und A5 enthält die Adresse des Hauptdatenbereichs von AMOS. Wenn Ihr Programm zu Ende ist, können Sie mit einer einzigen RST-Anweisung ins Basic zurückkehren.

Params ist eine Liste von Parametern, die durch den CALL-Befehl auf den A3-Stapel geschoben wird. Diese Parameter müssen in UMGEKEHRTER Reihenfolge entfernt werden. Also ist der letzte Wert, den Sie in diese Anweisung eingegeben haben, der erste auf dem Stapel. Je nach Art Ihrer Parameter werden die Werte in A3 in einem

der folgenden Formate dargestellt:

Integer-Variablen: Enthält ein langes Wort, das eine normale AMOS Ganzzahl-Variable aufweist.

Fließkomma-Variablen: Enthält eine Fließkommazahl im IEEE-Format mit einfacher Genauigkeit.

Zeichenketten-Variablen: Speichert die Adresse der Zeichenkette. Alle Zeichenketten beginnen mit einem Wort, das ihre Länge enthält.

Achtung! Niemals mit POKE direkt in eine Zeichenkette gehen! Wenn eine Zeichenkette zu einer Konstante initialisiert wird, so weist ihre Adresse auf die ursprüngliche Zuweisungsanweisung in dem aktuellen Programmlisting hin. Wenn Sie also diesen Wert ändern, hat das Auswirkungen auf den ursprünglichen Quelltext. Das ist natürlich äußerst ungünstig und sollte unbedingt vermieden werden.

In **BEISPIEL 21.1** im MANUAL-Unterverzeichnis finden Sie eine Demonstration von PLOAD und CALL.

=AREG= *(Variable, durch die Information auf die 68000er Adreßregister übertragen wird)*

a=AREG(r)
AREG(r)=a

AREG ist ein Array von sechs PSEUDO-Variablen, das eine Kopie der ersten sechs unter den 68000er Adreßregistern enthält. *r* kann zwischen 0 und 6 liegen und gibt die Nummer des Adreßregisters an, auf das diese Anweisung wirken soll.

Wenn ein CALL-Befehl ausgeführt wird, dann wird der Inhalt diese Arrays stets in die Adreßregister A0 bis A2 geladen. Am Ende werden Sie dann mit der neuen Information, die in die entsprechenden Register gestellt wurde, wieder gespeichert.

=DREG= *(Variable, durch die Information auf die 68000er Datenregister übertragen wird)*

d=DREG(r)
DREG(r)=d

Dies ist ein Array aus acht Integern, das eine Kopie des Inhalts der 68000er Datenregister enthält. *r* bezieht sich auf die Registernummer und kann zwischen 0 und 7 liegen, und so D1 bis D7 bezeichnen.

Zugang zu den Libraries des Systems

Mit AMOS können Sie auch die meisten der internen Amiga System-Libraries direkt aus

dem ROM aufrufen. Aber eigentlich ist das nicht besonders nützlich, denn alle wirklich interessanten Befehle haben wir bereits in AMOS eingebaut.

Sie sollten mit diesen Routinen nur dann arbeiten, wenn Sie wirklich auch genau wissen, was Sie hier tun. Der Amiga ist dafür bekannt, daß er schwer zu programmieren ist, und es ist nur allzu leicht, das System völlig abstürzen zu lassen und versehentlich den berüchtigten GURU-Fehler hervorzurufen.

=DOSCALL *(DOS-Library)*

r=DOSCALL(Funktion)

Führt eine Funktion direkt aus der DOS-Library aus. *Funktion* ist der Offset der entsprechenden Funktion. Weitere Einzelheiten dazu finden Sie im ROM-Kernel- Manual des Amiga.

Bevor Sie diese Funktion einsetzen, müssen Sie einige der Steuerregister in D0 bis D7 und A0 bis A2 mit den AREG- und DREG-Funktionen einstellen. Wenn die Routine ins Basic zurückkehrt, dann wird der Inhalt von D0 in *r* ausgewiesen. **Achtung:** Die Ergebnisse werden nicht in DREG und AREG geladen.

=EXECALL *(EXEC-Library)*

r=EXECALL(Funktion)

Ruft die EXEC-Library des Amiga auf. Zu Beginn werden D0 bis D7 und A0 bis A2 mit den Kontrollwerten aus den DREG- und AREG-Arrays geladen. *r* weist dann den Inhalt von D0 aus.

=GFXCALL *(Grafik-Library)*

r=GFXCALL(Funktion)

Ruft eine Routine aus der Grafik-Library unter Einsatz der in den DREG- und AREG-Arrays gespeicherten Kontrollwerte auf.

Funktion ist der Offset zu der betreffenden Funktion. *r* stellt das Ergebnis der Operation dar, das in D0 ausgewiesen wird.

=INTCALL *(Intuition-Library)*

r=INTCALL(Funktion)

Führt einen Befehl aus der Intuition-Library aus. Wie immer werden die Kontrollwerte aus den DREG- und AREG-Arrays geladen und r enthält das Ergebnis der Operation.

Da AMOS die normale Intuition-Routine nicht einsetzt, ist diese Funktion besonders gefährlich. Rufen Sie sie also nur auf, wenn Sie mit der Intuition-Library des Amiga schon vertraut sind.

Im Inneren von AMOS-Basic

Um den Entwicklern auch vollen Zugang zum Innenleben von AMOS-Basic zu bieten, haben wir einen “Einstieg” in die verschiedenen Datenbereiche eingebaut. Das ist allerdings nicht für den Durchschnitts-Programmierer gedacht, sondern nur dazu, daß fortgeschrittene Anwender ihre eigenen AMOS-Utilities erstellen können.

=SCREEN BASE *(Zeigt Schirmtabelle an)*

Tabelle=SCREEN BASE

Weist die Basisadresse der internen Tabelle aus, in der Position und Nummer Ihrer AMOS Schirme gespeichert werden. **BEISPIEL 21.2** bietet Ihnen eine einfache Demonstration.

=SPRITE BASE *(Zeigt Sprite-Tabelle an)*

Tabelle=SPRITE BASE (n)

Gibt die Adresse der internen Datenliste für das Sprite mit der Nummer *n* an. Wenn das Sprite nicht existiert, so lautet die Adresse der Tabelle 0.

Negative Werte für *n* ermitteln die Adresse der MASK, die Ihrem Sprite wahlweise zugewiesen werden kann. Nun weist *Tabelle* einen von drei möglichen Werten auf, je nach Status der Maske:

Tabelle<0	Zeigt, daß es für dieses Sprite keine Maske gibt.
Tabelle=0	Sprite *n* hat eine Maske, aber sie ist von dem System noch nicht generiert worden.
Tabelle>0	Dies ist die Speicheradresse der Maske. Das erste lange Wort dieses Bereichs enthält die Länge der Maske, und auf das nächste folgt die eigentliche Definition.

Siehe dazu auch **BEISPIEL 21.3** im MANUAL-Unterverzeichnis.

=ICON BASE *(Gibt die Adresse eines Icons an)*

Tabelle=ICON BASE(n)

Weist die Adresse für das Icon mit der Nummer n aus. Das Format dieser Information stimmt genau mit dem Format der weiter vorne dargestellten Funktion SPRITE BASE überein.

Die System-Requester-Erweiterung

REQUEST

REQUEST ON

Dies ist der Default, und AMOS generiert so seine eigene Requester-Routine.

REQUEST OFF

Wird dieser Befehl eingesetzt, so wählt AMOS stets die CANCEL-Taste des Requesters. Der Requester selbst wird nicht angezeigt, also eignet sich diese Funktion besonders gut zur Fehlersuche in einem Programm.

REQUEST WB

Hier wird AMOS angewiesen, auf den System-Requester der Workbench umzuschalten. Nachdem Sie dann eine der Optionen gewählt haben, kehren Sie zu AMOS zurück.

Achtung: Wenn Sie den Requester nicht laden (indem Sie ihn aus der Erweiterungsliste der Config-Datei löschen), werden alle Meldungen über den normalen Workbench-Requester angezeigt.

Das hat jedoch eine negative Nebenwirkung, denn man denkt, AMOS sei abgestürzt, wenn ein Requester erscheint. Sollte dies geschehen, drücken Sie einfach nur Amiga+A, um in die Workbench zurückzugehen, beantworten die Frage, drücken nochmals Amiga+A, um zu AMOS zurückzukehren. Am besten vermeiden Sie es, den Requester zu laden, wenn der Speicher sehr knapp wird!



23: AMOS Serielle Device-Erweiterung

Willkommen in der aufregenden Welt der AMOS-Kommunikation. Die serielle Erweiterung ist für alle diejenigen gedacht, die mit Hilfe der seriellen Schnittstelle des Amiga Informationen zwischen mehreren Computern austauschen möchten.

Mit der Erweiterung erhalten Sie im Ganzen 15 neue Befehle. Mit diesen Befehlen können Sie z.B. tolle Spiele für mehrere Spieler schreiben. Außerdem können Sie auf ein Midi-Interface, das an die serielle Schnittstelle des Amiga angeschlossen ist, zugreifen. So können die Musiker ihre Instrumente direkt aus AMOS-Basic heraus steuern. Und natürlich wird auch Multitasking voll unterstützt.

Sollten Sie Probleme mit der seriellen Schnittstelle haben, und ihr Computer nach dem ersten (einwandfreien) Benutzen der Schnittstelle unerklärlich abstürzt, so könnte es sein, daß Sie ein fehlerhafte Version des Serial.device von Commodore haben. Wenden Sie sich in diesem Fall mit Ihren Originaldisketten an den Customer Support von EuroPress Software. Die Adresse erfahren Sie durch das Drücken von HELP im AMOS Editor.

Das Öffnen der seriellen Schnittstelle

SERIAL OPEN *(Öffnet einen Kanal für das serielle I/O)*

SERIAL OPEN Kanal, Port_Nr [,Shared, Xdisabled, 7wires]

Öffnet einen Kommunikationskanal zu einem seriellen Device.

Kanal Das ist eine Identifikationsnummer, die in allen folgenden Kommunikationsbefehlen eingesetzt wird. Die zulässigen Werte liegen zwischen 0 und 3.

Port_Nr Gibt die logische Devicenummer der seriellen Schnittstelle an. Normalerweise sollte dieser Wert auf Null gesetzt werden. Wenn Sie jedoch eine MULTI-SERIAL-Karte in Ihren Amiga eingesetzt haben, können Sie über die Zahlen ab eins auf Ihre zusätzlichen Schnittstellen zugreifen.

Shared (wahlweise) Dieses Flag zeigt AMOS, daß das Device mit anderen Tasks, die gerade auf Ihrem Amiga ablaufen, geteilt werden kann. Man verwendet diesen Parameter beim Multitasking. Setzt man den Wert Null (FALSCH) ein, so wird der Kanal für AMOS-Basic belegt, und allen anderen Programmen der Zugriff verweigert. Wird -1 (RICHTIG) eingesetzt, so können sich mehrere im Speicher befindliche Programme die serielle Schnittstelle teilen. Achtung: Dieses System muß mit äußerster Vorsicht eingesetzt werden, sonst stürzt der Amiga leicht ab!

Xdisabled (wahlweise) Schaltet während der Übertragung Ihrer Daten über die serielle Schnittstelle die XON/OFF-Prüfung ein und aus. Auch wenn Sie diese Funktion erst später benötigen, müssen Sie dieses Flag setzen, wenn Sie die Schnittstelle für das Device öffnen. Der Default ist auf FALSCH (0) gestellt, und das System außer Kraft gesetzt. Durch den Wert -1 (RICHTIG) können Sie das System betriebsfähig machen.

Nach dem Öffnen der Schnittstelle müssen Sie dann durch Aufrufen des Befehls SERIAL X die XON- und XOFF-Zeichen einstellen.

7wires (wahlweise) Der Wert -1 (RICHTIG) weist das Device an, das 7-Leitungssystem entsprechend der offiziellen Commodore-Dokumentation einzusetzen. Der Default lautet FALSCH (0).

Wenn Sie den Befehl Serial Open zum ersten Mal anwenden, so wird automatisch die SERIAL.DEVICE-Library von Ihrer Systemdiskette geladen. Also stellen Sie zuerst sicher, daß sie sich auch im aktuellen Laufwerk befindet.

Als Default wird von AMOS automatisch eingesetzt :

Bei Port_Nr=0 ... die Einstellungen des Preferences-Programmes der Workbench. Falls Sie mit der Workbench arbeiten, sollte Sie Port_Nr=0 setzten.
Bei Port_Nr=2 ... hier werden die Einstellungen des französischen MiniTel genutzt : 1200 Baud, 7 bits, 1 stop bit, Even Parity.

Falls nötig, können Sie diese Einstellungen durch die Anweisungen SERIAL SPEED, SERIAL BIT oder SERIAL PARITY ändern.

Schließen der seriellen Schnittstelle

SERIAL CLOSE *(Schließt einen oder mehrere serielle Kanäle)*

SERIAL CLOSE [Kanal]

Wenn Sie die Kanalnummer nicht angeben, so schließt SERIAL CLOSE alle gegenwärtig geöffneten seriellen Kanäle, dabei wird jedoch keinerlei Fehlersuche durchgeführt. Durch das wahlweise Eingeben einer Kanalnummer können Sie einen bestimmten Kanal schließen, dann wird die übliche Fehlersuche durchgeführt.

Anmerkung: Wenn Sie mit RUN ein Programm aus AMOS-Basic heraus ablaufen lassen, so werden alle geöffneten Kanäle automatisch für Sie geschlossen.

Das Senden von Information über die serielle Schnittstelle

SERIAL SEND *(Ausgabe einer Zeichenkette über eine serielle Schnittstelle)*

SERIAL SEND Kanal, T$

Die Zeichenkette *T$* wird direkt durch den angegebenen seriellen Kanal geschickt. Dieser Befehl wartet die Übertragung der Daten über die tatsächliche Schnittstelle nicht ab. Deshalb müssen Sie durch die Funktion =SERIAL CHECK(Kanal) abfragen, ob die Übertragung abgeschlossen ist.

SERIAL OUT *(Ausgabe eines Speicherblocks über eine serielle Schnittstelle)*

SERIAL OUT Kanal, Adresse, Länge

Dieser Befehl stimmt mit SERIAL SEND überein, außer daß hier mit Ursprungsdaten gearbeitet wird.

Adresse stellt die Speicheradresse Ihrer Daten dar.

Länge ist Anzahl der Bytes, die gesandt werden sollen.

Das Einlesen von Information über die serielle Schnittstelle

SERIAL GET *(Erfaßt ein Byte über eine serielle Schnittstelle)*

=SERIAL GET (Kanal)

Ein einzelnes Byte wird über die serielle Schnittstelle eingelesen. Anmerkung: Der Wert -1 zeigt an, daß nichts verfügbar ist.

SERIAL INPUT$ *(Erfaßt eine Zeichenkette über die serielle Schnittstelle)*

=SERIAL INPUT$ (Kanal)

Liest eine ganze Zeichenkette über die serielle Schnittstelle ein. Sind keine Daten verfügbar, so erscheint eine leere Kette "". Andernfalls enthält die Zeichenkette alle Bytes, die bis zu diesem Zeitpunkt über die serielle Schnittstelle gesandt wurden.

Wenn Sie diesen Befehl mit der Hochgeschwindigkeitsübertragung (wie MIDI zum Beispiel) einsetzen, sollten Sie vorsichtig sein. Wenn Sie zwischen den einzelnen SERIAL INPUT$ Befehlen zu lange warten, so können Sie damit das System total überlasten, und störende Fehlermeldungen wie "Zeichenkette zu lang" oder "Serial device Puffer Überlauf" hervorrufen.

Das Einstellen der seriellen Parameter

SERIAL SPEED *(Stellt die Baudrate für einen seriellen Kanal ein)*

SERIAL SPEED Kanal, Baudrate

Stellt die aktuelle Übertragungsgeschwindigkeit für den betreffenden Kanal ein. Dieser Wert wird auf die Ein- wie auch die Ausgabe angewandt. Beachten Sie bitte, daß Sie für einen Kanal keine zwei verschiedenen Baudraten einstellen können.

Wenn die von Ihnen angegebene Baudrate von dem aktuellen Device nicht unterstützt wird, so kann sie vom System abgelehnt werden.

SERIAL BIT *(Stellt die Anzahl der Bits und Stop-Bits im Protokoll ein)*

SERIAL BIT Kanal, NBits, Stop-Bits

Zuweisung der Anzahl der Bits, die für jedes Zeichen, das Sie übertragen eingesetzt werden.

NBits ist die Anzahl der Bits.

Stop-Bits stellt die Anzahl der Stop-Bits dar.

SERIAL PARITY *(Stellt die Paritätsprüfung ein)*

SERIAL PARITY Kanal, Parität

Einstellen der Paritätsprüfung für den aktuellen seriellen Kanal. Nachstehend finden Sie die verfügbaren Optionen aufgeführt.

PARITÄT <0: Keine Parität
PARITÄT >0: Nur die ersten beiden Bits dieses Wertes sind von Bedeutung.

Bit 0: =0 ->GERADE
=1 ->UNGERADE

Bit 1: =0 ->Normale Parität
=1 ->LEERSTELLE

Das Paritätsbit kann folgendermaßen mit den Funktionen BSET oder BCLR eingestellt werden:

```
P=0 : Bset 0,P : Rem Ungerade Parität
Bclr 1,P : Rem Normale Parität
Serial Parity 1,P : Rem Einstellen der Parität durch den Wert in P
```

Eine ausführliche Erklärung dieses Systems finden Sie in der Commodore-Dokumentation.

SERIAL X *(Stellt XON/XOFF ein)*

SERIAL X Kanal, Xmode (Aktiviert/Deaktiviert das XON/XOFF-Handshake-System)

Der Wert RICHTIG für Xmode setzt das Handshake-System außer Kraft. Durch jeden anderen Wert wird es angestellt. Sie sollten die korrekten Steuerzeichen in Xmode laden. Diese müssen in folgendem Format eingegeben werden:

```
Xmode=XON*$10000000+XOFF*$10000
```

Weitere Einzelheiten hierzu entnehmen Sie bitte der Commodore-Dokumentation.

Sonstige Befehle

SERIAL BUFFER *(Stellt die Größe des seriellen Puffers ein)*

SERIAL BUFFER Kanal, Länge

Weist dem entsprechenden Kanal einen Puffer der in Länge festgelegten Anzahl von Bytes zu. Der Default ist auf 512 Bytes gestellt und die Mindestgröße beträgt 64 Bytes. Bei Hochgeschwindigkeitsübertragungen ist es ratsam, den Puffer zu erweitern.

SERIAL FAST *(Stellt den FAST-Übertragungsmodus ein)*

SERIAL FAST Kanal

Hier wird ein besonderes "fast" Flag im aktuellen Device gesetzt, und so eine Reihe von internen Tests außer Kraft gesetzt, die sonst den Kommunikationsprozeß verlangsamen würden. Setzen Sie diese Anweisung bei Hochgeschwindigkeitsübertragungen wie MIDI ein. Achtung: Wenn Sie diesen Befehl aufrufen, wird das Protokoll auf GERADE PARITÄT, KEIN XON/OFF, 8 Bits geändert.

SERIAL SLOW *(Stellt die serielle Übertragung auf langsam zurück)*

SERIAL SLOW Kanal

Stellt das serielle Device auf die normale, langsamere Geschwindigkeit zurück und aktiviert die Fehlersuche wieder.

SERIAL CHECK *(Weist den aktuellen Status des seriellen Device aus)*

=SERIAL CHECK (Kanal)

Fragt den aktuellen Status des seriellen Device ab. Mit dieser Anweisung können Sie überprüfen, ob die gesamte Information, die Sie durch den Befehl SERIAL SEND übertragen wollten, gesandt wurde.

CHECK=FALSE (0) ->	wenn der letzte SERIAL-Befehl noch ausgeführt wird
CHECK=TRUE (-1) ->	Alles erledigt!

SERIAL ERROR *(Zeigt Erfolg oder Mißerfolg der letzten Übertragung an)*

=SERIAL ERROR (Kanal)

Fragt das FEHLER-Byte des seriellen Device ab. Der Wert Null zeigt an, daß die Übertragung zufriedenstellend erfolgte. Andernfalls verlief die letzte Übertragung fehlerhaft.

Das Senden von langen Zeichenketten

Die Übertragung von langen Zeichenketten kann - vor allem bei niedrigen Baudraten - etwas zeitraubend sein. Da AMOS multitasking-fähig ist, läuft Ihr Programm jedoch nach einer SERIAL SEND Anweisung weiter ab.

Bevor die Übertragung beendet ist, sollte die "Müllabfuhr", das Freisetzen von verfügbarem Speicher vermieden werden, denn sonst werden Ihre Daten zerstört.

Also:

- Setzen Sie die Funktion =SERIAL CHECK ein, bevor Sie viel mit Zeichenketten arbeiten.
- Führen Sie mit dem Befehl X=FREE die Müllabfuhr durch, um sicherzustellen, daß Ihr Programm sie nicht automatisch auslöst.
- Verwenden Sie die Anweisung SERIAL OUT Kanal, Adresse, Länge, wobei "Adresse" die Adresse der zuvor reservierten Speicherbank enthält.

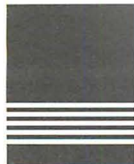
Weitere Information

Weitere Informationen über das serielle System des Amiga finden Sie im Commodore-Handbuch ROM KERNEL Reference Manual, Library and Devices. Auf diese Art können die Experten unter Ihnen die seriellen Devices zu größtmöglichem Nutzen einsetzen.

Der Quelltext

Den Quelltext für die Erweiterung finden Sie auf der AMOS-Aktualisierungsdiskette (APD36) zusammen mit der AMOS Basic-Gleichungsdatei. Zur Erstellung wurde Genam einsetzt. Sie brauchen alle ".I"-Dateien des Commodore und die AMIGA.LIB- Datei, um sie zu programmieren.

Arbeiten Sie mit "dependant" Symbol-Case, "linkable" Programmtyp und "export" Debug-Info. Erstellen Sie die Verbindung mit BLINK SERIAL.O bis SERIAL.LIB AMIGA.LIB.



24: Das Konfigurieren von AMOS

AMOS kann leicht auf Ihre persönlichen Anforderungen zugeschnitten werden, angefangen beim Umstellen der Fehlermeldung auf eine andere Sprache, bis zum Konfigurieren des Bildschirms zum Einsatz mit einem NTSC-System. Alle Optionen, die AMOS laden und ausführen kann, können vom Konfigurationsprogramm (Config.AMOS) aus geändert werden.

Das AMOS-Konfigurationsprogramm

Eine der interessantesten Eigenschaften von AMOS-Basic ist die Möglichkeit, das Paket genau auf Ihre speziellen Anforderungen zuschneiden zu können. Durch die AMOS Konfigurations-Utility erhalten Sie die bisher noch nie dagewesene Kontrolle über das Innenleben des AMOS-Systems. Es gibt eine große Anzahl möglicher Optionen, durch die Sie alles Erdenkliche ändern können, von der Höchstanzahl der BOBs auf dem Bildschirm bis zum genauen Erscheinungsbild der verschiedenen Menüfenster. Sie können sogar den Text der Fehlermeldungen, die im Laufe Ihres Programms erscheinen, völlig verändern! All diese Informationen sind in einer besonderen Konfigurationsdatei im AMOS Systemverzeichnis gespeichert. Diese Daten werden stets automatisch geladen, wenn AMOS läuft.

Da diese Utility so leistungsstark ist, muß man sie jedoch mit etwas Vorsicht behandeln. In der Praxis kann ein schwerer Fehler nämlich dazu führen, daß Ihr ganzes System unbrauchbar wird! Wir würden Ihnen deshalb sehr stark raten, all Ihre Änderungen auf der SICHERUNGSKOPIE der AMOS-Programmdiskette zu machen. Verwenden Sie niemals die AMOS Originaldiskette zu diesem Zweck! Wenn alle Stricke reißen, senden wir Ihnen gerne eine neue Konfigurationsdatei, aber Sie können inzwischen ja nicht mit AMOS-Basic arbeiten.

Das Laden der Konfigurations-Utility

Leichter kann das Laden der Konfigurations-Utility eigentlich garnicht sein: legen Sie die SICHERUNGSKOPIE Ihrer AMOS-Programmdiskette in das interne Laufwerk und laden Sie die Datei Config1_3.Amos. Nach dem Einstieg in das Programm müssen Sie dann angeben, ob Sie den Editor-Puffer erweitern wollen. Hier klicken Sie dann JA an, oder drücken die J-Taste.

Jetzt können Sie wie üblich mit dem Programm arbeiten. Nach der Initialisierung erscheint ein hübscher Regenbogeneffekt auf dem Bildschirm. Die verschiedenen Menüoptionen können Sie nun durch Drücken der rechten Maustaste heraufholen.

Bevor es nun weitergeht, müssen Sie erst eine der bestehenden Konfigurationsdateien in den Speicher laden. Die Befehle LADE/SICHER finden Sie im Disk-Menü. Wenn Sie zum ersten Mal mit diesem Programm arbeiten, wählen Sie die Option Lade Grund-Konfiguration, damit die ursprünglichen Parameter von der Diskette eingegeben werden.

Nachdem Sie dann alles etwas verändert haben, können Sie diese Änderungen

durch die Optionen Sicher Konfiguration oder Sicher Als ... auf die Diskette speichern. Wenn Sie dann das nächste Mal AMOS-Basic laden, werden diese Parameter automatisch geladen. Hier ist eine vollständige Liste der verschiedenen Menüoptionen:

AMOS-Menü

Über dieses Programm: Zeigt einen einfachen Titelschirm an.

Ende: Zurück zu AMOS-Basic. Achtung! Wenn Sie Ihre Konfiguration nicht zuvor auf Diskette gespeichert haben, gehen alle Änderungen verloren.

AMOS verlassen: Verläßt AMOS-Basic und geht ins CLI oder zur Workbench zurück.

Disk-Menü

Lade Grund-Konfiguration: Lädt die vorhandene Konfigurationsdatei von der AMOS-Programmdiskette. Sie ist unter dem Dateinamen AMOS1_3_PAL.Env (bzw. AMOS1_3_NTSC.Env bei NTSC-Systemen) im Verzeichnis AMOS_System gespeichert.

Lade andere Konfiguration: Lädt eine andere Konfigurationsdatei von der Diskette. Es ist völlig zulässig, mehrere dieser Dateien auf der Bootdiskette zu haben, aber nur die AMOS1_3_PAL.Env Defaultdatei hat tatsächlich eine Wirkung.

Wenn Sie ein CLI-Fan sind, dann können Sie sich bestimmt eine kleine Batch-Datei schreiben, durch die Sie während des Start-Ups eine von mehreren verschiedenen Konfigurationen auswählen können. In diese Batch-Datei sollten Sie dann im Verlauf Ihrer normalen Startup_sequence einsteigen können. Vorausgesetzt, Sie bringen all Ihre Konfigurationen im AMOS_System-Verzeichnis unter, müssen Sie dann nur noch die ursprüngliche Konfigurationsdatei zum Beispiel in AMOS.EN1 umbenennen, und der Datei, die Sie ausgewählt haben, den Namen AMOS1_3_PAL.ENV zuweisen.

Sicher Konfiguration: Speichert die aktuellen Parameter auf Ihre Programmdiskette. Diese Parameter werden jetzt beim Laden von AMOS-Basic stets im Speicher installiert. Verändern Sie die Konfiguration auf der Original-Programmdiskette NIEMALS, oder Ihr Setup könnte hoffnungslos durcheinanderkommen!

Sicher als ...: Speichert Ihre Parameter in einer eigenen Datei. Die aktuelle Konfiguration bleibt dabei völlig unverändert.

Einstellen-Menü

Neue Tastatur: Durch diesen Befehl wird der von AMOS-Basic eingesetzte Tastaturtreiber verändert. Diese Treiber MÜSSEN mit dem speziellen Keyboard- Definer auf der AMOS-Programmdiskette erstellt werden. Die Treiber auf der Amiga Extras-Diskette sind hier NICHT zulässig. Sie finden im KEYBOARDS-Verzeichnis auf der AMOS-Programmdiskette eine Liste der derzeit verfügbaren Treiber.

Geladene Erweiterungen: Anders als viele Mitbewerberprodukte kann AMOS-Basic unwahrscheinlich leicht erweitert werden. Ganze Libraries an neuen Anweisungen

können über ein leistungsstarkes System an Erweiterungsdateien direkt an das vorhandene Paket angefügt werden. Wenn Sie in der Maschinensprache programmieren können, so können Sie diese sogar selbst erzeugen! Weitere Einzelheiten können Sie dem Quelltext auf der AMOS Extras-Diskette entnehmen.

Wenn Sie Ihrem System eine neue Erweiterungsdatei hinzufügen, so müssen Sie AMOS die genaue Adresse auf der Diskette mitteilen. Das geschieht über die AMOS-Erweiterungsliste. Gehen Sie einfach mit dem Cursor auf die entsprechende Stelle und klicken Sie die linke Maustaste an. Nun können Sie den vollständigen Pfadnamen Ihrer Erweiterung über die Tastatur eingeben, und mit ENTER abschließen. Wie üblich können alle Zeichenketten über die normalen Cursor-Tasten beliebig überarbeitet werden.

Help Accessory Name: Durch diesen Befehl können Sie Namen und Position des geladenen Zusatzes verändern, wenn Sie im Editor die Hilfe-Taste drücken. Die ersten beiden Zeichenketten enthalten den Namen Ihres neuen Hilfe- Zusatzprogramms. Diese beiden Zeichenketten müssen stets gleich sein, sonst erhalten Sie eine Fehlermeldung, wenn Sie versuchen, dieses System anzuwenden.

Der letzte Punkt auf dieser Liste enthält die Meldung, die angezeigt wird, wenn das Zusatzprogramm nicht auf der aktuellen Diskette gefunden wird.

AMOS System Datei Namen: Liefert Ihnen ein Liste von wichtigen Systemdateien, auf die das AMOS-Paket zugreift. In diesen Dateien sind die Daten enthalten, die für die AMOS Programm-Icons, Maus-Cursor, Fonteinstellungen und Tastaturdefinitionen erforderlich sind. Durch das Ändern dieser Definitionen können Sie das Erscheinungsbild vieler Ihrer Basic-Programme radikal verändern.

Port Namen: Enthält eine vollständige Liste aller Devices, die von AMOS-Basic aus angesprochen werden können. Wir haben bereits die große Mehrzahl aller Devices miteinbezogen. Wenn Sie allerdings etwas Ungewöhnliches, wie zum Beispiel Midi-Interfaces hinzufügen, müssen Sie ihre Bezeichnung in diese Liste eintragen. Sie müssen dann auch die entsprechenden Device-Treiber im Devs- Verzeichnis hinzufügen.

Funktionstasten: Durch diese Funktion können Sie jede der vom AMOS-Editor eingesetzten Menüoptionen umbenennen. Gehen Sie einfach mit der Maus über die Option und klicken Sie die linke Taste an. Jetzt können Sie den neuen Namen des gewählten Befehls eintippen. Wenn Deutsch z.B. nur Ihre 2. Sprache ist, so können Sie AMOS an Ihre Muttersprache anpassen. Und wenn Sie einfach nur ein bißchen Spaß haben möchten, haben Sie hier genug Freiraum für etwas kreativen Blödsinn.

Die Wirkung dieser Menübefehle bleibt völlig unverändert, ganz egal, wie Sie sie nennen!

Negativ-Verzeichnis Filter: Enthält eine Liste von Dateien, die automatisch bei allen zukünftigen Directory-Listings wegfallen. Diese Dateien können falls erforderlich Platzhalter (auch Wildcards genannt) enthalten, und so können leicht eine ganze Reihe von Dateien gleichzeitig unterdrückt werden. Für weitere Einzelheiten siehe den Befehl AMOS SET DIR.

Amiga Funktionstasten: AMOS-Basic enthält ein leistungsstarkes System an Tastaturmakros. Zugang zu diesen Makros enthält man durch Drücken der rechten oder linken Amiga-Taste in Verbindung mit einer der verschiedenen Funktionstasten. Weitere Einzelheiten dazu finden Sie unter dem Befehl KEY$.

Als Default werden für diese Makros eine Reihe von häufig verwendeten Basic-Anweisungen eingesetzt. Sie können sie jedoch praktisch beliebig verändern.

Editor Einstellung: Diese Optionen bestimmen die Art, auf die AMOS Dateien auf Diskette speichert, und verändern sowohl die Größe des Puffers wie auch die Scroll-Geschwindigkeit des Editierfensters. Wenn Sie beabsichtigen, AMOS von der Workbench aus einzusetzen, so möchten Sie wahrscheinlich ein Icon zusammen mit Ihren Basic-Programmen speichern. Eine weitere nützliche Option ist die Erweiterung der Größe des vom Editor eingesetzten Textpuffers. So wird die maximale Länge der Basic-Programme, die in Ihren Computer eingegeben werden können, erhöht.

Screen Anzeige Einstellung: Bestimmt Position und Größe der verschiedenen Bildschirmbereiche wie zum Beispiel Editierfenster, Menükasten und Direkt-Modus-Schirm. In der Praxis ist es am klügsten, diese Funktion mit angemessener Vorsicht einzusetzen, da nicht alle Optionen in gleichem Maße zulässig sind. Sogar AMOS hat seine Grenzen, und wenn Sie versuchen, das Bildformat bis zur Unkenntlichkeit zu verändern, so funktioniert AMOS wahrscheinlich nicht mehr.

Interpreter Einstellung: Wenn Sie genug Speicherplatz zur Verfügung haben, können Sie durch diese Optionen die Anzahl der BOBs erhöhen und die Größe des Sprite-Puffers verändern (siehe SET SPRITE).

Sie können auch den Speicherbereich erweitern, der von der anwenderdefinierten Copperliste eingesetzt wird. Und so erhalten Sie dann Zugang zu einer großen Reihe von Spezialeffekten.

Screen Farben - Seite 1: Verändert alle Farben, die von den AMOS Editierfenstern eingesetzt werden. Vor der Anpassung dieser Parameter ist es jedoch ratsam, die ursprünglichen Werte durch die Option Sicher Als... auf einer neuen Diskette zu speichern. Auf diese Weise können Sie dann nach Herzenslust experimentieren, ohne in Gefahr zu laufen, eine völlig unbrauchbare Anzeige zu erhalten.

Wie Sie wahrscheinlich schon vermuteten, wird die Intensität des Farbwertes durch die +-Tasten gesteigert und durch die -Tasten reduziert. Jedesmal, wenn Sie die Maus auf die entsprechenden Optionen stellen, wird eine sichtbare Anzeige der aktuellen Farbeinstellungen erstellt.

Screen Farben - Seite 2: Verändert die Farben, die vom Direkt-Modus-Fenster, der Zeile für die Fehlermeldung und dem Follow-Schirm eingesetzt werden.

AMOS als NTSC System: Diese Option justiert alle Bildschirmanzeigen so, daß sie der NTSC-Bildschirmgröße entsprechen. Das ist besonders nützlich für Anwender in Amerika, da AMOS generell auf PAL eingestellt ist.

AMOS als PAL System: Stellt alle Optionen für die Bildschirmanzeige für die PAL-Konfiguration ein.

Verändere Mitteilungen

Durch dieses Menü können Sie alle vom AMOS-System generierten Meldungen durch Ihre eigene, verbesserte Version ersetzen. Alle Optionen in diesem Menü funktionieren auf ähnliche Weise: Zum Editieren einer Meldung stellen Sie den Maus-Cursor auf die entsprechende Zeile und klicken die linke Taste an. Nun können Sie Ihre neue Zeichenkette direkt über die Tastatur eingeben und den Text mit den normalen Cursor-Tasten editieren. Wenn Sie dann mit der Arbeit fertig sind, können Sie die neue Meldung durch Drücken der ENTER-Taste installieren.

Links vom Meldungsfenster befinden sich drei Icons. Mit den Pfeil-Icons können Sie vorwärts und rückwärts durch die Meldungszeilen scrollen. Durch die Option Quit kehren Sie in den Konfigurationshauptschirm zurück.

Vom Editor: Diese Meldungen werden vom AMOS Editor generiert und in der Statuszeile angezeigt. Gegen Ende der Liste finden Sie eine Reihe von Textketten, die das allgemeine Erscheinungsbild der Statuszeile definieren. Auch dies können Sie nach Belieben verändern.

Während des Programm-Tests: Jedesmal, wenn Sie ein Programm ablaufen lassen, oder den TEST- Befehl anklicken, führt AMOS eine eingehende Überprüfung der Basic-Syntax Ihres Programmes durch. Treten hier irgendwelche Probleme auf, so erscheint die entsprechende Fehlermeldung. Auch diese Fehlermeldungen können wie oben beschrieben verändert werden.

Während des Programm-Starts: In der Praxis können jedoch nicht alle Fehler durch die Eingangsprüfung der Syntax ermittelt werden. Deshalb gibt es eine getrennte Reihe von Run-Time-Meldungen, die generiert werden, wenn AMOS einen Fehler entdeckt, während Ihr Programm tatsächlich abläuft.

Von allen LADE/SPEICHER Zugriffen: Das ist eine ziemlich bunte Mischung von Text, der vom Editor während der Lade- und Speicher-Operation ausgegeben wird. Manche der Meldungen werden in der Statuszeile angezeigt, während andere hingegen die Titel von verschiedenen Datei-Requestern darstellen.

Außerdem gibt es eine Reihe von Zeichenketten, die die Suchpfade für die Load-, Save- und Merge-Befehle definieren. Wenn Sie zum Beispiel Programme im ASCII-Format aus STOS oder Amiga Basic importieren, kann es günstig sein, den Pfadnamen von *.ASC auf *.* zu ändern.

Vom Dateiauswahlfenster: Wollen Sie sich den Datei-Requester maßschneidern? Wenn ja, dann liegen Sie bei dieser Option ganz richtig! Alle Tasten und Meldungen können vom Konfigurationsprogramm aus beliebig verändert werden. Sie können sogar die Pfeile, die für die Scroll-Balken verwendet werden, austauschen!

Vom Fehler Bildschirm: Das sind die Meldungen, die in der Fehlermeldungszeile angezeigt werden, wenn in Ihrem Basic-Programm ein Problem auftritt.

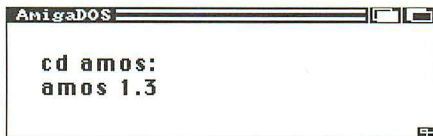
Vom Direkt Modus: Bestimmt das allgemeine Erscheinungsbild des Funktionstastenfensters, das im Direkt-Modus angezeigt wird.

Vom AMOS Titelbild: Jedesmal, wenn AMOS in den Speicher geladen wird, wird in der Mitte des Editierfensters ein Titelbildschirm angezeigt. Durch diese Option können Sie den Titel ganz nach Ihren eigenen Vorstellungen gestalten.

Beachten Sie dabei, daß AMOS automatisch eine vollständige Liste aller gegenwärtig installierten Erweiterungen an Ihren neuen Titelbildschirm anfügt.

Das Aufrufen von AMOS aus dem CLI

Falls nötig kann man mit AMOS auch vom CLI aus arbeiten. Legen Sie einfach die Sicherungskopie Ihrer AMOS-Programmdiskette in ein Laufwerk und tippen Sie ein:



Es gibt drei mögliche Ausführungen von diesem Befehl:

AMOS1.3
Ruft AMOS-Basic auf und geht nach dem Startup in den Editor.

AMOS1.3 Dateiname
Ruft AMOS auf, lädt das gewählte Basic-Programm und läßt es ablaufen. Der Puffer des Editors ist normalerweise auf die erforderliche Mindestgröße gestellt, um so Speicherplatz zu sparen. Dazu folgendes Beispiel:

amos1.3 map_editor.amos (Läßt nach dem Laden von AMOS den Map Editor ablaufen)

AMOS1.3 -Speicher
Lädt AMOS-Basic und reserviert die entsprechende Anzahl von Bytes für den Editor. Beachten Sie hier bitte das Minuszeichen vor der Zahl. Als Default ist der Editor auf eine Größe von 32.767 KByte gestellt. Dies kann entweder durch die Option Textpuffer aus dem AMOS Such-Menü, über eine Option im CONFIG- Programm oder direkt von der Befehlszeile aus erhöht werden. Beispiel:

amos1.3 -50000 (Ruft AMOS auf und reserviert 50 KByte für den Editor)

AMOS kann auch durch eine Batch-Datei oder als Teil der normalen Startup- Sequenz im S: Verzeichnis auf Ihrer Startdiskette aufgerufen werden.

Das Aufrufen von AMOS aus der Workbench

Starten Sie die Workbench wie üblich und legen Sie die AMOS-Programmdiskette in ein beliebiges Laufwerk. Wählen Sie die AMOS: Diskette an, indem Sie mit der Maus eine Liste der verfügbaren Programmdateien anzeigen lassen. Durch Doppelklicken im entsprechenden Icon können Sie jetzt direkt in AMOS-Basic einsteigen.

Achtung! Dieses Vorgehen ist bei einem A500 ohne Speicher-Erweiterung nicht empfehlenswert, da die Workbench unwahrscheinlich viel Speicher frißt und Ihnen so fast nichts für Ihre Basic-Programme übrigläßt. Es ist wesentlich günstiger, den Amiga mit der Programmdiskette im internen Laufwerk zu booten, und AMOS so zu laden.

Das Laden eines AMOS-Programms aus der Workbench

Jedem Ihrer AMOS Basic-Programme kann direkt ein Icon zugewiesen werden. Diese Funktion kann durch eine spezielle Option in der Konfigurations-Utility aktiviert werden. Dann wird in der Folge für jedes Basic-Programm, das Sie auf der Diskette speichern, ein Icon erzeugt. Wenn Sie nun dieses Icon von der Workbench aus wählen, so wird AMOS automatisch in den Speicher geladen und Ihr Basic- Programm läuft sofort ab.

Beachten Sie dabei aber bitte, daß Anwender mit Festplatte erst etwas Konfigurationsarbeit leisten müssen, bevor sie dieses System einsetzen können. Klicken Sie das entsprechende Programm-Icon mit der linken Maustaste an und wählen Sie die Option INFO aus dem Workbench-Menü. Nun verändern Sie das DEFAULT TOOL dahingehend, daß es auf die Kopie von AMOS-Basic weist, die Sie auf Ihrer Festplatte installiert haben. Sie sollten dieses Vorgehen für jedes Programm, das Sie von der Workbench aus ablaufen lassen möchten, wiederholen.

Das Unterbringen von AMOS an einer beliebigen Stelle auf der Festplatte/Diskette

AMOS übernimmt eine Reihe von Verwaltungsaufgaben, bevor das Laden beendet ist und Sie mit der Arbeit beginnen können. Eine dieser Aufgaben besteht darin, daß alle Defaultdateien wie zum Beispiel der Sprite-Zeiger, Defaultfont usw. lokalisiert werden. Dabei wird folgendermaßen verfahren:

1 Das aktuelle System wird überprüft, um festzustellen, ob ein PAL- oder NTSC-Bildschirm eingesetzt wird.
2 Je nach System wird im aktuellen Verzeichnis entweder nach der Datei AMOS1_3_PAL.Env oder AMOS1_3_NTSC.Env gesucht.
3 Kann keine dieser Dateien gefunden werden, so wird das Verzeichnis AMOS_System im Root-Directory des aktuellen Laufwerks gesucht. Schlägt dies ebenfalls fehl, so gibt AMOS den Ladeversuch auf und kehrt in die Workbench zurück.
4 Stößt AMOS am Punkt 2 auf die Defaultdatei, so wird die Environment-Datei untersucht, um zu ermitteln, wo sich die anderen Dateien befinden. Auf diese Art kann man AMOS an jeder beliebigen Stelle auf der Diskette/Festplatte unterbringen.

Nehmen wir an, daß wir AMOS in einem Verzeichnis namens UTILS auf der Festplatte unterbringen wollen. Zuerst müssen wir dazu die folgenden Dateien von der AMOS-Programm-Diskette in den Pfad DH0:UTILS/ auf der Festplatte kopieren.

AMOS_System (Ein Verzeichnis, das alle Defaultdateien enthält)
AMOS1.3 (Das AMOS-Hauptprogramm)

Die Dateien AMOS1_3_PAL.Env und AMOS1_3_NTSC.Env, die sich im

AMOS_System-Verzeichnis befinden, müssen aus diesem Verzeichnis herausgeholt und in das UTILS-Verzeichnis gestellt werden.

AMOS_System
AMOS1.3
AMOS1_3_PAL.Env
AMOS1_3_NTSC.Env

Nun müssen wir die Default.Env-Dateien verändern, so daß sie AMOS mitteilen, wo alle Datendateien zu finden sind. Dazu müssen Sie eine normale Kopie von AMOS booten und das Config1_3.AMOS-Programm ablaufen lassen. Jetzt müssen Sie nur einfach sicherstellen, daß die geladenen Erweiterungen und Defaultdateien über die richtigen Pfad- und Dateinamen verfügen.

In unserem Beispiel befinden sich alle Dateien in dem Verzeichnis AMOS_System. Die Erweiterungen müssen nun folgendermaßen lauten:

1-AMOS_System/Music
2-AMOS_System/Compact
3-AMOS_System/Request
4-
5-
6-AMOS_System/Serial

Die Namen der Defaultdateien müssen so aussehen:

1-AMOS_System/Def.Icon
2-AMOS_System/Mouse.Abk
3-AMOS_System/Default.Font
4-AMOS_System/Default.Key

Wichtig ist dabei, daß hier nur der zusätzliche Pfadname erforderlich ist. AMOS fügt AMOS_System dem aktuellen Pfad (in diesem Fall Dh0:UTILS) hinzu, um den vollständigen Pfad Dh0:UTILS/AMOS_System zu erhalten.

Diese Information ist für Anwender, die mit Festplatte arbeiten, sehr nützlich und Sie können so vermeiden, daß das AMOS-System-Verzeichnis im Root-Directory Ihrer Festplatte landet. Außerdem wird diese Funktion auch für alle, die eine CDTV-Applikation in AMOS entwickeln, sehr nützlich sein.

Die obengenannten Punkte treffen ebenfalls auf das RAMOS-Runtime-System zu. Der einzige Unterschied besteht darin, daß RAMOS die PAL- und NTSC-Dateien nicht benötigt.



25: RAMOS Runtime-System

RAMOS ist eine eigenständige Run-Only-Version (läuft nur ab, verfügt nicht über Editor etc.), mit der Sie Ihre Programme völlig unabhängig vom AMOS- Softwarepaket ablaufen lassen können.

Im Unterschied zu AMOS-Basic ist RAMOS ein Public-Domain-Programm, und kann deshalb beliebig oft mit jedem Ihrer Basic-Programm verteilt werden. Auf diese Art können Sie also Ihre Programme ganz legal an jeden verkaufen, der einen Amiga hat, auch wenn dieser nicht über AMOS-Basic verfügt.

Was Ihre Programme betrifft, so ist RAMOS praktisch identisch zu AMOS-Basic. Es gibt nur ein paar Unterschiede:

- Es gibt keinen Editor! RAMOS vollzieht einen automatischen Close Editor Befehl nach dem Laden. Tritt ein Fehler auf, oder wird Ctrl+C gedrückt, so geht RAMOS direkt in die Workbench zurück, ohne zuvor eine Fehlermeldung anzuzeigen.

- Es gibt keinen Defaultschirm. Auf diese Art können Sie Ihre Schirme zu Beginn Ihres Programms initialisieren, ohne zuerst den Defaultschirm anzeigen zu müssen. Wenn Ihr Programm den Defaultschirm selbst einsetzt, so müssen Sie am Anfang Ihres Listings die folgende Zeile hinzufügen:

  Screen Open 0,320,200,16,Lowres

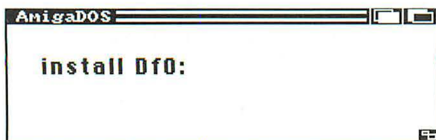
Das Installieren von RAMOS auf einer neuen Diskette

Um Ihre Programme im Run-Only-Format verteilen zu können, müssen Sie erst eine spezielle Run-Only-Diskette vorbereiten. Dazu gehen Sie folgendermaßen vor:

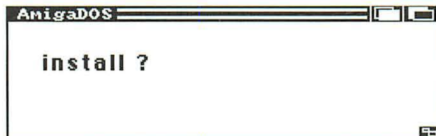
- Fertigen Sie eine Kopie Ihrer Original AMOS Program Disk an, indem Sie ein handeslübliches Kopierprogramm nutzen, oder die Workbench. Näheres entnehmen Sie bitte Ihrem Commodore Handbuch.

- Laden Sie das RAMOS Installierungsprogramm von der AMOS Extras- Diskette (RAMOS1_3_Install.Amos) und lassen Sie es ablaufen. Nun folgen Sie einfach den Anweisungen, um RAMOS auf Ihre neue Diskette zu kopieren.

- Zum Schluß verlassen Sie AMOS und gehen ins CLI. Mit dem INSTALL- Befehl machen Sie Ihre Diskette startfähig. Zum Beispiel:



Wenn Sie auf ein einziges Laufwerk beschränkt sind, so müssen Sie den Befehl folgendermaßen eingeben:



Nach der nächsten Aufforderung legen Sie Ihre neue Diskette ins Laufwerk und tippen:

Df0:

Das Erzeugen einer RAMOS-Startdiskette

Speichern Sie Ihr Programm auf der Run-Only-Diskette unter dem Namen AUTOEXEC.AMOS. Jedesmal, wenn Sie Ihren Computer mit dieser Diskette starten, wird automatisch RAMOS geladen und Ihr Basic-Programm läuft ab.

Das Aufrufen eines RAMOS-Programms aus der Workbench

Bevor Sie in diese Funktion einsteigen können, müssen Sie AMOS erst anweisen, ein Icon für Ihre Programme zu erzeugen, wenn sie auf Diskette gespeichert werden. Das ist ganz einfach und muß nur einmal durchgeführt werden.

- Laden Sie eine SICHERUNGSKOPIE von AMOS-Basic und lassen Sie die CONFIG1_3.AMOS-Utility wie üblich ablaufen.
- Nachdem das Programm initialisiert wurde, laden Sie die Defaultparameter von der Diskette mit der Option Lade Konfiguration aus dem Disk- Menü ein.
- Nun rufen Sie die Funktion ‘Editor Einstellung’ aus den Einstellung-Menü auf und klicken die Option “Sicher Icons” an.
- Gehen Sie ins Hauptmenü zurück und wählen Sie Sicher Konfiguration. Die neuen Parameter werden automatisch aktiviert, wenn Sie nun AMOS-Basic laden.

Sie können Ihre Programme jetzt direkt auf die Run-Only-Diskette speichern. Wenn Sie in der Folge in die Workbench einsteigen, so hat nun jedes Programm sein eigenes Icon im Directory-Fenster.

Der letzte Schritt besteht darin, RAMOS als Default-Tool zu speichern. Klicken Sie das entsprechende Programm-Icon an und wählen Sie die Option INFO aus dem Workbench-Menü. Nun ändern Sie das Default-Tool von :AMOS auf :RAMOS, und installieren Sie das Icon mit der Option Save auf der Diskette.

Nachdem Sie diese Schritte ausgeführt haben, können Sie nun Ihre Programm direkt aus der Workbench durch doppeltes Anklicken des gewünschten Icons laden. Wie üblich können Sie zwischen Ihrem Programm und der Workbench jederzeit durch Drücken der Tastenkombination Amiga+A hin- und herspringen.

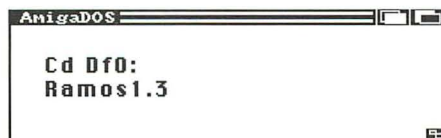
Zugang zu RAMOS aus dem CLI

Wenn Sie sich im Amiga CLI Zuhause fühlen, können Sie direkt aus der Befehlszeile Zugang zu RAMOS erhalten. Die Syntax lautet folgendermaßen:



Programm.amos stellt dabei das Programm dar, das Sie ablaufen lassen möchten. Wenn Sie diesen Parameter nicht angeben, so wird RAMOS versuchen, eine Datei namens AUTOEXEC.AMOS aus dem aktuellen Verzeichnis zu laden.

Falls erforderlich, kann RAMOS auch als Teil einer Batch-Datei geladen werden. **Hinweis**: Stellen Sie vor dem Aufrufen von RAMOS stets das aktuelle Laufwerk mit dem CD-Befehl auf die Run-Only-Diskette, sonst kann RAMOS die verschiedenen Systemdateien, die erforderlich sind, damit das Programm laufen kann, nicht finden. Zum Beispiel:



Anmerkungen zu RAMOS

Die von RAMOS verwendeten Konfigurationsparameter können Sie in der Datei RAMOS.ENV im Verzeichnis AMOS_System finden. Diese Datei enthält keine der üblichen Fehlermeldungen, deshalb ist sie ungefähr 7 KByte kleiner als die entsprechende AMOS-Datei. Sie können diese Parameter mit der CONFIG1_3.AMOS-Utility modifizieren, die sich auf der AMOS-Programmdiskette befindet.

Sollte Ihr Programm Befehle oder Funktionen einer AMOS Erweiterung wie AMOS

3D oder TOME nutzen, so laden Sie die Default-Konfigurationsdatei mit dem CONFIG1_3.AMOS- Utility und notieren Sie sich die Eintragungen im 'Geladene Erweiterungen' Menupunkt. Laden Sie nun RAMOS.ENV und ändern Sie die Eintragungen im 'Geladene Erweiterungen' Menupunkt auf die vorher notierten Werte. Nur wenn Sie dies getan haben, kann RAMOS ihr Programm richtig ausführen. Sollte es auf einen Befehl treffen, der zu einer nicht richtig eingetragenen, und somit nicht richtig geladenen Erweiterung gehört, so bricht RAMOS sofort ab.

Achtung! Als Default wird RAMOS automatisch versuchen, beim Laden jedesmal die Workbench zu schließen. Dadurch ist es nicht möglich, nach erfolgreichem Abschluß Ihres Programms zur Workbench oder ins CLI zurückzukehren. Diese Schwierigkeit können Sie vermeiden, indem Sie die entsprechende Option in der Konfigurationsdatei RAMOS.ENV verändern. Durch die Option Editor Setup im Konfigurationsprogramm kann diese Änderung durchgeführt werden. Nähere Informationen hierzu können Sie der Dokumentation zur AMOS Konfigurations- Utility entnehmen.



26: Zusatzprogramme (Accessories)

Der AMOS Sprite-Editor

Wie Sie wissen, bietet Ihnen AMOS-Basic zahlreiche, leistungsstarke Eigenschaften für die Manipulation von BOBs und Sprites. Um diese Objekte in Ihren Spielen einzusetzen, brauchen Sie natürlich eine Möglichkeit zum Erzeugen dieser verschiedenen Sprite-Bilder auf Ihrem Computerbildschirm. Und genau das ist die Rolle des AMOS Sprite-Editors. Außerdem können Sie mit diesem Programm auch Icons erzeugen. Zeichnen Sie einfach Ihre Objekte als BOBs und speichern Sie sie auf Diskette. Nun können Sie Ihre Objekte mit der Utility Icon_Conv.Amos auf der AMOS-Programmdiskette direkt in Icons umwandeln.

Die durch diesen Editor erzeugten Bilddateien funktionieren übrigens mit Sprites und BOBs gleich gut. Der einzige wesentliche Unterschied zwischen diesen Objekten besteht darin, daß Sprites auf höchstens 16 Farben beschränkt sind. Zur Vereinfachung beziehen wir uns im folgenden Text immer auf Sprites anstatt Sprites/BOBs.

Das Laden des Sprite-Editors

Der Sprite-Editor kann entweder als normales Programm oder als Zusatzprogramm geladen werden.

Das Laden als Programm

Legen Sie Ihre SICHERUNGSKOPIE der AMOS-Programmdiskette ins aktuelle Laufwerk und laden Sie die Datei Sprite_Editor.Amos. Nun klicken Sie Run an und nach einer kurzen Pause läuft der Sprite-Editor ab.

Der Einsatz des Editors als Zusatzprogramm

Den AMOS Sprite-Editor können Sie als Zusatzprogramm einsetzen, indem Sie ihn über die Option Andere Lade im Befehlsfenster (Shift/F7) laden. Nachdem er in den Speicher geladen wurde, erhalten Sie mit der Option St. Andere aus dem Editor direkt Zugang zum Sprite-Editor. Auf Ihr aktuelles Programm hat dieses Vorgehen überhaupt keine Auswirkungen.

Übrigens dauert das Initialisieren des Sprite-Editors ungefähr 10 Sekunden. Während dieser Zeitspanne sehen Sie sich manchmal einem leeren Bildschirm gegenüber. Kein Grund zur Panik, das ist ganz normal! Nach ein paar Sekunden können Sie dann mit dem Sprite-Editor arbeiten.

Das Arbeiten mit dem Sprite-Editor

Die meisten Funktionen des AMOS Sprite-Editors werden über eine Reihe von einfachen Schirm-Icons gesteuert. Aufrufen können Sie diese Funktionen durch Anklicken der gewünschten Option mit der linken Maustaste. Die einzigen entscheidenden Ausnahmen zu dieser Regel sind die Befehle, mit denen Bildbanken geladen und gespeichert werden.

Durch die **L**-Taste wird eine AMOS Sprite-Bank in den Speicher geladen. Danach erscheint ein normaler AMOS Datei-Requester auf dem Bildschirm, über den eine Sprite-Bank wie üblich ausgewählt und geladen werden kann.

Entsprechend können Sie Ihre neuen Sprite-Bilder über die **S**-Taste auf der aktuellen Diskette speichern. Beachten Sie dabei aber bitte, daß dieser Editor nur Sprite-Banken im AMOS .ABK Standardformat laden kann.

Die Steuerung

Der Bildschirm ist in vier Hauptbereiche aufgeteilt:

Das Icon-Fenster Dieses Fenster ist in 5 Zeilen zu 10 Icons aufgeteilt. Jedes Icon steuert eine Funktion des Sprite-Editors. Um eine dieser Optionen aufzurufen, muß man das entsprechende Icon mit der linken Maustaste anklicken.

Das Editierfenster Der Hauptbereich für das Zeichnen befindet sich links im Editierkasten. Das Zeichnen eines Bildes könnte garnicht einfacher sein. Man geht einfach mit dem Maus-Cursor in den Editierkasten und hält eine der Maustasten gedrückt. Jeder Maustaste kann eine eigene Farbe aus dem Palettenfenster zugewiesen werden. Wenn Sie zeichnen und dabei beide Tasten gleichzeitig gedrückt halten, wechselt die Farbe willkürlich zwischen diesen beiden Farben ab. Während Sie Ihr Objekt erstellen, wird das Bild im Sichtfenster automatisch aktualisiert.

Das Sichtfenster Dieses Fenster enthält eine Kopie des Sprites, das Sie gerade zeichnen, in seiner Originalgröße. Die Dimensionen dieses Objektes können jederzeit verändert werden, indem man den Sichtbereich mit der linken Maustaste vergrößert. Um die neue Einstellung zu aktivieren, klicken Sie einmal mit der rechten Maustaste.

Die Farbpalette Die Farbpalette enthält eine Auflistung der verfügbaren Farben, die Sie in Ihrer Zeichnung einsetzen können. Um die Farbe zu ändern, klicken Sie die gewünschte Schattierung entweder mit der linken oder der rechten Maustaste an. Die neue Farbe wird jetzt jedesmal dann eingesetzt, wenn Sie im Editierfenster die entsprechende Taste gedrückt halten.

Sollte die Palette den Farbton, den Sie wünschen, nicht aufweisen, so können Sie die gegenwärtige RGB-Einstellung durch das RGB-Icon in der linken unteren Ecke des Icon-Fensters verändern.

Die Zeichen-Icons (Dritte und vierte Zeile von oben)

Plot

Zeichnet einzelne Punkte an der aktuellen Mausposition.

Zeichnen

Dem Befehl Plot sehr ähnlich, abgesehen davon, daß die Punkte, die Sie auf den Bildschirm zeichnen, durch eine Linie verbunden werden. Der resultierende Effekt ist im allgemeinen wesentlich glatter.

Linie

Zieht eine gerade Linie zwischen zwei beliebigen Punkten. Der Ausgangspunkt wird durch Drücken der linken Maustaste festgelegt. Wenn Sie nun diese Taste gedrückt halten, während Sie die Maus bewegen, können Sie eine Linie mit der gewünschten Länge ziehen. Den Endpunkt der Linie können Sie durch die linke Taste an jede beliebige Stelle setzen. Durch Drücken der rechten Maustaste kann der Vorgang jederzeit abgebrochen werden.

Pinsel

Um den Cursor werden willkürlich Punkte angeordnet. Durch Drücken der linken Maustaste kann der Pinsel auf die gewünschte Größe ausgedehnt werden. Die Leistung des Pinsels kann durch eine Option der Funktion Glätten geregelt werden.

Rechteck

Wie bei dem Befehl Linie bestimmen Sie auch hier zuerst die Form durch Verschieben der Maus bei gedrückter linker Taste. Dann lassen Sie die Taste los und bewegen das Objekt an die gewünschte Stelle. Durch einmaliges Anklicken mit der linken Maustaste wird das Rechteck positioniert. Wenn Sie die Arbeit beendet haben, können Sie den Vorgang durch Drücken der rechten Maustaste abbrechen. Nun können Sie ein weiteres Rechteck zeichnen oder durch Anklicken der Linien- oder Plot-Icons zum Zeichen-Modus zurückkehren.

Rechteck Voll

Identisch zum normalen Rechteck-Befehl, außer daß hier ein durch das aktuelle Füllmuster ausgefülltes Rechteck erzeugt wird. Das Füllmuster kann jederzeit durch die Icons + und - verändert werden.

Ellipse

Nicht gefüllte Ellipse. Die Dimensionen für diese Objekt werden dadurch bestimmt, daß Sie den Maus-Cursor ziehen und gleichzeitig die linke Maustaste gedrückt halten. Nachdem Sie Ihre Ellipse definiert haben, können Sie sie durch einmaliges Klicken mit der linken Maustaste auf dem Bildschirm positionieren. Wie üblich wird der Vorgang mit der rechten Maustaste abgebrochen und die Abmessungen der Ellipse auf Null zurückgesetzt.

Text

Sie können in dem derzeitig definierten Font Ihrem Sprite etwas Text hinzufügen. Der Text kann direkt über die Tastatur eingegeben werden. Mit der Rücktaste (Backspace) kann er korrigiert, und durch Klicken der linken Maustaste positioniert werden.

Füllen

Füllt unregelmäßige Formen auf. Wird die linke Taste gedrückt, so dauert das Füllen solange an, bis eine Farbe erreicht wird, die sich von Ihrer Füllfarbe unterscheidet (Überfluten).

Alle Balken- und Füll-Funktionen greifen auf die derzeit definierten Füllmuster zu. Die Füllmuster können durch Anklicken der Icons + und - geändert werden. Beachten Sie auch, daß Sie bei Rechteck, Balken, Ellipse, Linie, Text und Pinsel durch Anklicken der rechten Maustaste die Größe verändern können.

Cut/Paste (Ausschneiden/Einsetzen)

Links oben auf dem Bildschirm befinden sich zwei Icons, die es Ihnen ermöglichen, jeden beliebigen Teil des Editierfensters auszuschneiden und in an einer neuen Stelle im aktuellen Sprite wieder einzusetzen. Das Ausschneiden wird durch das Scherensymbol dargestellt.

Wenn Sie diesen Befehl gewählt haben, müssen Sie zuerst die Maus an die linke obere Ecke des Bereichs stellen, den Sie erfassen möchten. Dann halten Sie die linke Maustaste gedrückt und markieren so den gewünschten Bereich. Wenn Sie die Taste loslassen, wird dieser Bereich zur weiteren Verwendung als Pinsel in den Speicher geladen. Um den Bereich zu kopieren, klicken Sie einfach mit der linken Maustaste die neue Position an. Durch die rechte Maustaste wird der Bereich unter dem Block durch das aktuelle Füllmuster ersetzt.

Wenn Sie das Einsetzen beendet haben, können Sie eines der Zeichen-Icons anklicken und an Ihrem Design weiterarbeiten. Der gegenwärtige Block wird im Speicher sicher aufbewahrt und kann durch Auswählen des Paste-Icons wieder aktiviert werden.

Rechts von den Icons für das Ausschneiden und Einsetzen ist das Transparenz-Icon. Hier können Sie wählen, ob während des Einsetzens die Farbe Null auf den Bildschirm gezeichnet werden soll.

Mit der nächsten Gruppe von Icons können Sie die Position des “Griffs” des derzeit gewählten Pinsels verändern. Diese Icons können in Verbindung mit den Befehlen Rechteck, Balken, Linie, Ellipse und Paste (Einsetzen) angewendet werden.

Hot-Spot

Unter den Icons für das Ausschneiden und Einsetzen ist eine Gruppe von Icons, die die Position des Hot-Spots Ihres Sprites bestimmen. Der Hot-Spot dient als Referenzpunkt für die Berechnung der Position Ihres Sprites auf dem Bildschirm. Wenn er also in der Mitte des Sprites liegt und das Sprite auf die Koordinaten 23,40 gestellt wird, so befindet sich auch die Mitte des Sprites bei 23,40.

Die ersten sieben Icons bewegen den Hot-Spot auf eine von mehreren festgelegten Positionen. Die verschiedenen Optionen lauten von links nach rechts: Oben links, Oben Mitte, Oben rechts, Unten links, Unten Mitte, Unten Rechts, Mitte.

Mit dem letzten Icon können Sie den Hot-Spot beliebig auf dem Sprite positionieren. Wenn der Cursor auf den Hot-Spot Icons steht, wird stets der aktuelle Hot-Spot im Editierfenster angezeigt.

Die Größe des Sprites

Rechts oben im Icon-Fenster finden Sie vier Icons, durch die Sie das Format des Sprites, das Sie gerade bearbeiten, verändern können. Das erste, mit der Bezeichnung See as Sprite, zeigt Ihnen das aktuelle Bild genauso, wie es durch den Befehl AMOS SPRITE angezeigt werden würde. (Sprites sind schneller als BOBs, aber auf 16 Farben beschränkt und verfügen noch über andere Bildschirmbegrenzungen. Weitere Einzelheiten entnehmen Sie bitte dem AMOS Handbuch).

Die anderen drei Icons sind:

Größe Zeigt einen Cursor-Block an, mit dem Sie dann die Größe des Sprites verändern können. Der Zoom-Faktor des Editierfensters stellt sich automatisch auf das Optimum für die jeweils gewählte Größe ein. Drücken Sie einmal auf die rechte Maustaste, um diesen Modus zu verlassen. Die Größe des Sprites kann auch verändert werden, indem Sie den Anzeigebereich mit der linken Maustaste ziehen und die rechte Taste drücken, wenn das Sprite die erforderliche Größe hat.

Optimieren Schiebt das Sprite und schrumpft die Größe dann auf das für das Bild erforderliche Minimum (spart so Speicher).

Schieben Das Bild wird so weit wie möglich nach links oben geschoben.

Anmerkung: Wenn ein Sprite innerhalb einer Größe von 32 Pixel in beide Richtungen liegt, so wird es vollständig in vierfacher Vergrößerung dargestellt. Über diese Größe hinaus wird ein Fenster von 32x32 Pixel in vierfacher Vergrößerung dargestellt. Über die Cursor-Tasten kann der angezeigte Bereich verändert werden. Der Bereich, den Sie bearbeiten, wird durch zwei Markierungsbalken im Sprite- Anzeigebereich dargestellt.

Weitere Editierfunktionen

Rechts von den Zeichen-Icons befinden sich mehrere nützliche Funktionen zur Bearbeitung des Bildes.

LÖSCHEN Löscht das Bild unter Verwendung der aktuellen Füllmuster/Farben.

H/V Flip Flippt das Bild horizontal oder vertikal.

Rotieren Rotiert das Bild 90 Grad im Uhrzeigersinn.

Scroll Scrollt das Bild im Editierfenster (zum Aussteigen rechte Taste).

Zoom Sie können das Bild verkleinern und vergrößern. In diesem Modus verändert das Anklicken des Anzeigekastens mit der linken Maustaste die Größe des Bildes. Drücken der rechten Taste fixiert das Bild in der aktuellen Größe. Wie üblich erfolgt der Abbruch durch Auswählen eines anderen Zeichen-Modus.

SCHIRM Hier können Sie Bildschirmauflösung und Anzahl der Farben, mit denen Sie gerade arbeiten, verändern. Der Editor unterstützt gegenwärtig folgende Schirm-Modi:

Low Res (Niedrige Auflösung): 8,16,32 und 64 (EHB) Farben
High Res (Hohe Auflösung): 2, 4, 8 und 16 Farben

Das 'Klauen' von Sprites

Der Transfer des Objektes, das Sie bearbeiten, von und zu der Sprite-Bank kann über eine Gruppe von Icons unten auf dem Bildschirm erfolgen. Links von diesen Icons ist die RGB-Taste, über die Sie die RGB-Werte aller Farben im Palettenfenster verändern können. Nach der RGB-Taste sind die Greif-Icons in folgendermaßen angeordnet:

Sprite einfügen Schafft Platz und stellt das Sprite dann in die Bank.

Sprite setzen Stellt das aktuelle Bild direkt in die Sprite-Bank, wobei das Sprite, das zuvor dort war, überschrieben wird.

Sprite holen Holt ein Sprite aus der Bank, das Sie dann bearbeiten können.

Sprite löschen Löscht ein Sprite aus der Sprite-Bank (Befehl kann nicht durch Aufheben rückgängig gemacht werden!)

Bank radieren Löscht ALLE Sprites in der Bank (Befehl kann nicht durch Aufheben rückgängig gemacht werden!)

Unten rechts im Icon-Fenster sind fünf Icons, mit welchen Sie alle im Speicher verfügbaren Sprites durchblättern können.

I<	Springt zum ersten Sprite im Speicher.
<	Springt zur vorhergehenden Sprite-Definition.
Nummer	Geht in den Sichtmodus und zeigt jeweils drei Sprites gleichzeitig an. Aus diesem Modus kann man einfach durch Verschieben des Cursors vom Icon-Bereich weg aussteigen.
>	Zeigt das nächste Sprite in der aktuellen Liste an.
>I	Springt zum letzten Sprite im Speicher.

Während der Cursor auf einem dieser Icons steht, wird das ausgewählte Sprite im Sichtfenster angezeigt. Dieses Sprite hat auf das Objekt, das Sie gerade bearbeiten, keine Auswirkungen.

Sonstige Icons

Aufheben	Macht den letzten Zeichen-/Zoom-/Rotier-/Dreh- usw. -schritt rückgängig.
Exit	Steigt aus dem Programm aus.
Glätten	Hier können Sie die Rahmenfarben des Editierfensters, den "Ping"- Ton, der ertönt, wenn Sie ein Icon anklicken, und seine Lautstärke verändern. Außerdem können Sie die Fließgeschwindigkeit des Pinsels verändern.

Abkürzungstasten

Als Alternative zu den normalen Icons ist es auch möglich, eine Reihe von Operationen direkt über die Tastatur auszuführen. Wenn Sie die Shift-Taste gedrückt halten und gleichzeitig die Cursor-Tasten drücken, so können Sie damit folgendes ausführen:

Shift + oben	Speichert das Bild an der aktuellen Position in der Bank (entspricht der Taste SETZEN).
Shift + unten	Holt das aktuelle Sprite aus der Bank (entspricht der Taste HOLEN).
Shift + links	Speichert das Bild in der aktuellen Position und holt dann das vorhergehende Sprite aus der Bank.

Shift + rechts Speichert das Bild in der aktuellen Position und holt dann das nächste Sprite aus der Bank.

Zusätzlich dazu erhalten Sie über die Funktionstasten schnell Zugang zu den nächsten 10 Sprites in der aktuellen Speicherbank. Wenn nun zum Beispiel der Sprite-Zeiger auf Sprite 5 gestellt ist, so geht F1 zu Sprite 5, F2 zu Sprite 6 usw. bis zu F10. Die Funktionstaste F10 greift dann auf Sprite 14 zu.

Durch Drücken von Shift in Verbindung mit einer Funktionstaste kann man die Sprites in der Bank speichern. Laden kann man das Sprite, indem man nur die Funktionstaste drückt.

Überlegungen zum Speicher

Das Programm funktioniert auf einem A500 ohne Erweiterung perfekt. Wenn Sie unbedingt mehr Speicher benötigen, um eine große Anzahl von Sprites zu erzeugen, so können Sie zum Beispiel die Glätt-Routinen (aber nicht die Tasten- Routinen) entfernen. Sie können auch etwas Speicher sparen, indem Sie die Picture-Bank-Routine (Bank 6) modifizieren.

Im vom AMOS Club herausgegebenen Newsletter erhalten Sie ausführliche Information darüber, wie Sie das Beste aus AMOS auf einer 1/2 Megabyte- Maschine herausholen können. In der ersten Ausgabe finden Sie einen Artikel mit dem Titel "A500 Blues". Wenn Sie also bisher noch kein Mitglied des AMOS Clubs sind, so sollten Sie sofort beitreten!

Der Sprite-Grabber

Durch den Sprite-Grabber können Sie eine Reihe von Sprite-Bildern aus jedem beliebigen Bild im IFF-Format aufgreifen. Das bedeutet, daß Sie Ihre Objekte mit Ihrer Lieblings-Grafiksoftware erstellen und sie dann direkt in einer normalen AMOS Speicherbank speichern können.

Am Anfang des Programms sehen Sie einen AMOS Datei-Requester. Hier können Sie ein Bild auswählen, das von der aktuellen Diskette geladen werden soll.

In der Anwendung ist der Sprite-Grabber überraschend einfach. Alle Optionen werden durch einen schmalen Menü-Balken gesteuert, der sieben Icons enthält. Dieser Balken kann mit den Cursor-Tasten über das Bild nach oben bewegt werden. Sie können mit diesen Tasten auch den im Anzeigefenster sichtbaren Teil Ihres Bildes scrollen.

Eine Option können Sie einfach durch Anklicken des entsprechenden Icons mit der linken Maustaste auswählen. Hier folgt nun eine umfassende Erläuterung der verschiedenen Menü-Icons.

-/+ Zur Wahl der Nummer des Bildes, das Sie aufgreifen möchten. Wenn das Bild bereits definiert wurde, so wird eine Kopie am Menübalken entlang angezeigt. Achtung! Aufgrund eines kleinen Bugs in der gegenwärtigen Version des Sprite-Grabbers kann die Anzeige des Bildes auf dem Bildschirm versehentlich unterlassen werden. Dieses Problem löst man, indem man in den Direkt-

Modus zurückkehrt und das Grabber-Programm sofort nochmals aufruft. Ihre neuen Bilder werden durch dieses Manöver nicht im Geringsten beeinflußt.

Scheren-Icon

Erfaßt einen rechteckigen Bildschirm-bereich und lädt ihn in das aktuelle Bild. Wenn Sie diese Option wählen, verschwindet der Menü-Balken unverzüglich, und Sie erhalten Zugang zum gesamten Bildschirmbereich.

Sie können nun das Bild mit einem normalen Rechteck markieren, und den markierten Bereich in den Speicher kopieren. Stellen Sie nur den Cursor in die linke obere Ecke Ihres Bildes und halten Sie die linke Maustaste gedrückt. Wenn Sie die Maus über den Bildschirm bewegen, wird ein leerer Kasten um das aktuelle Bild gezeichnet. Wenn Sie mit Ihrem neuen Bild zufrieden sind, lassen Sie die Maustaste los, und das Bild wird in den Speicher geladen.

Diskette zu Schirm

Lädt ein Bild im IFF-Format von der Diskette.

Diskette zu .ABK

Lädt eine bestehende AMOS Sprite-Bank von der Diskette. Achtung! Diese Option zerstört Ihre vorhandenen Bilder vollständig. Beachten Sie auch, daß eine neue Farbpalette direkt von der Sprite-Bank geladen wird.

.ABK zu Diskette

Speichert Ihre Bilder in Form einer normalen Sprite-Bank auf Diskette. Diese Bank kann in der Folge direkt in eines Ihrer AMOS Basic-Programme geladen werden.

Exit

Kehrt vom Grabber zu AMOS-Basic zurück.

Der Total AMOS Map Editor (TAME)

Wenn Sie ein Spielhallenspiel schreiben, müssen Sie für Ihr Programm oft viele verschiedene Hintergrundschirme erzeugen. Die Erfahrung hat gezeigt, daß die effizienteste Art, diese Schirme zu erzeugen, darin besteht, sie aus einer Reihe von kleineren Bausteinen oder Versatzstücken zusammenzusetzen. So können Sie jeden Schirm einfach als eine Auflistung seiner Bestandteile speichern, und so ein beträchtliches Maß an Speicherplatz sparen.

Um nun einen dieser Schirme zu erzeugen, brauchen Sie offensichtlich eine Art Map-Editor. Das ist der Zweck des **T**otal **A**MOS **M**ap **E**ditors (TAME). Es handelt sich hier um eine leistungsstarke Zeichen-Utility, die es Ihnen ermöglicht, die verschiedenen Schirme, die Sie für Ihre Spiele benötigen, ganz mühelos erstellen. Ein ideales Beispiel, wie diese Maps umgesetzt werden können, finden Sie in dem ausgezeichneten Spiel Magic Forest auf der AMOS Datendiskette.

Nachdem Sie Ihre Maps einmal erzeugt haben, können Sie sie mit Hilfe der speziellen TAME-Prozeduren am Ende der Magic Forest Demo auf dem Bildschirm darstellen. In Zukunft werden Sie sie dann auch direkt mit dem in Kürze verfügbaren TOME-Erweiterungssystem anzeigen können.

Beim Startup fordert Sie der Map-Editor auf, ein Bild im IFF-Format über einen normalen Datei-Requester auszuwählen. Dieses Bild wird in den Speicher geladen und in eine Reihe von Versatzstücken aufgeteilt, die dann dazu verwendet werden, Ihre Maps zu erzeugen. Die Versatzstücke werden - begonnen in der linken oberen Ecke des Bildes - von links nach rechts zugewiesen.

Als Default wird die Map stets durch eine Kopie des Versatzstückes mit der Nummer Null aufgefüllt. Wenn Sie mit einem leeren Bildschirm beginnen wollen, sollten Sie sicherstellen, daß die linke obere Ecke Ihres Quellbildes völlig leer ist.

Der TAME-Schirm ist in drei Hauptbereiche aufgeteilt.

Die Menüzeile

Die Menüzeile wird durch ein großes A oben auf dem Bildschirm angezeigt. Wählt man diese Zeile mit der rechten Maustaste an, so wird eine Reihe von leistungsstarken Menüoptionen aufgerufen. Diese bieten eine Vielzahl nützlicher Eigenschaften für das Definieren Ihrer Maps.

Wird die Maus auf die Menüzeile gestellt, so erscheint ein Statusfenster auf dem Bildschirm.

Das Statusfenster

Das Statusfenster enthält eine Vielzahl von nützlichen Einzelheiten über Ihre Map:

Stück Enthält eine Kopie des Versatzstückes, das als Pinsel gewählt wurde. Beachten Sie aber bitte, daß die Farben dieses Versatzstückes sich von denen, die auf dem Bildschirm tatsächlich eingesetzt werden, stark abweichen können.

SCHIRM Zeigt Bezeichnung und Art des aktuellen Schirms an.

MAP Der AMOS Map-Editor ist völlig dazu imstande, Maps zu bearbeiten, die wesentlich größer als der sichtbare Schirmbereich sind. Alle Zeichenoperationen finden in einen Sichtfenster der aktuellen Map statt.

Rechts von der Statuszeile ist ein großes Rechteck, das einen grünen Balken enthält. Dabei handelt es sich eigentlich um einen Menüpunkt, der manipuliert werden kann, um die genaue Position des Sichtfensters in Bezug zur gesamten Map zu steuern. Halten Sie einfach die linke Maustaste gedrückt und ziehen Sie den Balken mit der Maus. Nun wird das Sichtfenster entsprechend verschoben. Die Map kann auch durch einen an der linken Buchse angeschlossenen Joystick oder durch Drücken der Pfeiltasten gescrollt werden.

O Lks — X: Enthält die horizontale Koordinate des aktuellen Sichtfensters in Bezug auf die linke obere Ecke der Map. Die Maßeinheit für diese Koordinaten stellen die einzelnen Versatzstücke dar.

Y: Gibt die Y-Koordinate des Teils der Map an, den Sie derzeit bearbeiten.

Größe — Zeigt die Gesamtgröße Ihrer Map an. X: Breite der Map in Versatzstücken. Y: Höhe der Map in Versatzstücken.

Das Zeichenfenster

Das Zeichenfenster nimmt den größten Teil des sichtbaren Bildschirmbereichs ein. Als Default wird das aktuelle Versatzstück stets der Maus zugewiesen. Es kann durch einfaches Anklicken der linken Maustaste an jeder beliebigen Stelle auf dem Bildschirm eingesetzt werden.

Die Auswahl eines Versatzstückes ist denkbar einfach: Stellen Sie die Maus auf das Zeichenfenster und drücken Sie die rechte Maustaste. Die Map wird unverzüglich durch den Titelschirm ersetzt. Sie können nun die Maus auf das gewünschte Versatzstück stellen und mit der linken Maustaste ins Zeichenfenster zurückkehren.

Die Menüoptionen

File-Menü

Stücke laden — Nun können Sie einen neuen IFF-Schirm laden, aus dem Sie dann Versatzstücke bilden können. Sie können auch Zugang zu einem Bild erhalten, das zuvor mit dem AMOS SPACK-Befehl gepackt wurde.

Map laden — Lädt eine neue Map aus einer .MAP-Datei auf der Diskette.

Map speichern — Speichert die aktuelle Map als .MAP-Datei.

Exit — Verläßt den Map-Editor und kehrt zu AMOS-Basic zurück.

Das Block-Menü

Block ausschneiden — Erfaßt einen Teil Ihres Schirms und speichert ihn in einem temporären Menübereich. Zuerst klicken Sie den **T L** Zeiger in der linken oberen Ecke des auszuschneidenden Bereichs an, dann stellen Sie den **B R** Zeiger auf den diagonal gegenüberliegenden Punkt. Sie können diesen Bereich nun ganz einfach durch Anklicken der linken Maustaste ausschneiden.

Block einsetzen Setzt einen zuvor erstellten Block auf dem Bildschirm ein. Stellen Sie den Cursor auf das Zeichenfenster und klicken Sie die linke Maustaste an, um den Block an der gewünschten Stelle einzusetzen.

Das Änderungsmenü

Stück Größe Ermöglicht es Ihnen, die Größe der Versatzstücke zu verändern. Mögliche Abmessen umfassen 16x16, 16x32, 32x16 und 32x32.

Map Größe Stellt die Größe der aktuellen Map in Einheiten von einzelnen Versatzstücken ein.

Das Edit-Menü

Zeichnen Halten Sie die linke Maustaste gedrückt, um ein Versatzstück an der aktuellen Cursor-Position zu zeichnen.

Rechteck Wie bei dem Befehl Ausschneiden, geben Sie auch hier die Abmessungen Ihres Rechtecks durch Anklicken der oberen und unteren Ecken des Bereichs an. Er wird dann mit dem aktuellen Versatzstück aufgefüllt.

Stück aussuchen Ermöglicht es Ihnen, aus dem aktuellen Satz ein Versatzstück auszusuchen. Sie können diese Option im Map-Fenster durch die rechte Maustaste auswählen.

Map löschen Löscht die Map entweder bis zum ersten Versatzstück im Satz oder dem derzeit ausgewählten Versatzstück.

Der Menü-Editor

Eine der charakteristischen Eigenschaften von AMOS-Basic ist das leistungsstarke Menüsystem. Es enthält viele Funktionen, die sogar bei einer Grafik-Workstation höchster Qualität beeindrucken würden.

Unglücklicherweise ist es leicht, beim Definieren der Menüs von der schieren Vielfalt der verfügbaren Befehle überwältigt zu werden. So besteht die starke Versuchung, Komplikationen vermeiden zu wollen und die Menüs so einfach wie möglich zu gestalten.

Der AMOS Menü-Editor umgeht dieses Problem, indem er es Ihnen ermöglicht, Ihre Menüs direkt mit Hilfe der Maus zu erstellen. Diese Menüs können dann in Form einer AMOS Speicherbank auf Diskette gespeichert werden. Sie können diese Bank dann in Ihre Basic-Programme laden und durch eine einzige Basic-Anweisung Zugang zum gesamten Menü erhalten.

Mit Hilfe des AMOS Menü-Editors wird auch das Experimentieren leicht gemacht. Sie können also problemlos ganz umwerfende Menüs in Ihre Programme einbauen. Der

Menü-Editor wird durch eine Reihe von einfachen Schirm-Icons gesteuert und alle Optionen können durch Auswahl der entsprechenden Taste und einmaliges Anklicken mit der linken Maustaste ausgeführt werden.

Das Laden des Menü-Zusatzprogramms

Der Menü-Editor kann entweder als Zusatzprogramm oder als normales Basic-Programm geladen werden. Wenn er als Zusatzprogramm installiert ist, so greift er automatisch auf alle beliebigen Icons, BOBs oder Menüs, die von Ihrem aktuellen Programm eingesetzt werden, zu.

Das System-Menü

Nach dem Startup erscheint sofort das Hauptsystemmenü. Es unterstützt eine Reihe von nützlichen Systembefehlen für das Laden, Speichern und Initialisieren Ihrer Menüs. Wie bereits erwähnt, sind alle Option für Sie zum Greifen nahe: stellen Sie einfach nur den Zeiger auf das entsprechende Icon und wählen Sie es durch Drücken der linken Maustaste an.

Hier ist eine detaillierte Aufstellung der Systembefehle:

Neues Menü erstellen

Definiert einen neuen Schirm als Hintergrund für Ihre Menüs. Das Format dieses Schirm kann über die Maus eingegeben werden. Dann können Sie in das Haupteditierfenster gehen, um die verschiedenen Menüpunkte zu erzeugen.

Vorhandenes Menü bearbeiten

Springt direkt in den Editierschirm und initialisiert das gegenwärtig ausgewählte Menü.

Menübank speichern

Erzeugt eine Speicherbank für Ihre Menüs und speichert sie dann auf der Diskette. Dieses Menü kann zum Beispiel mit einer Zeile wie:

```
Load "filename.abk"
```

in Ihr Programm geladen werden.

Sie können Ihr neues Menü mit:

```
Bank to menu 6
```

initialisieren. Dieses Menü kann jetzt durch den Befehl MENU ON aktiviert werden.

Beachten Sie bitte, daß es sich hier um permanente Menübanken handelt, die automatisch zusammen mit Ihren Basic-Programmen gespeichert werden. Wenn Sie also einmal Ihr Menü installiert haben, und den Menü- Handler in AMOS-Basic geschrieben haben, so brauchen Sie keinen Gedanken mehr daran zu verschwenden.

Menü laden

Lädt eine Menüdefinition aus einer .MENU-Datei von der Diskette. Diese Datei muß zuvor durch die Option Aktuelles Menü speichern definiert worden sein. Nachdem das Menü erfolgreich geladen wurde, springt das Programm sofort zum Haupteditierschirm.

Aktuelles Menü speichern

Speichert Ihre gesamte Menüdefinition in einer sequentiellen .MENU-Datei. Diese Datei kann nur vom Menü-Editor aus geladen werden. Wenn Sie von AMOS-Basic aus auf Ihr Menü zugreifen möchten, sollten Sie stattdessen die Option Menübank speichern verwenden.

Exit und Zugreifen

Rückkehr zu AMOS-Basic. Wenn Sie den Editor als Zusatzprogramm laufen lassen, so wird die neue Menübank automatisch installiert. Sie können nun direkt aus Ihrem aktuellen Programm auf das Menü, das Sie erzeugt haben, zugreifen.

Exit

Rückkehr zur AMOS-Basic. Achtung! Durch diesen Befehl wird Ihre Menüdefinition vollständig ausradiert!

Das Editierfenster

Das Editierfenster ist zweigeteilt. Oben auf dem Bildschirm wird das Menü, an dem Sie gerade arbeiten, angezeigt. Jeder Menüpunkt wird dabei durch ein kleines numeriertes Rechteck dargestellt. Diese Rechtecke sind folgendermaßen angeordnet:

- Alle Punkte auf einer bestimmten Menüstufe sind in derselben vertikalen Spalte angeordnet. So stellt die erste Spalte die Titelzeile Ihres Menüs dar, und die zweite die verschiedenen Menüpunkte.
- Jeder Punkt ist streng aufgrund seiner Stellung in der Menüstruktur numeriert. Zum Beispiel stellt Punkt 1.2 die zweite Menüoption des Titels mit der Nummer 1 dar.

Die oben beschriebene Anordnung mag zwar auf den ersten Blick etwas kompliziert erscheinen, aber mit ein wenig Übung können Sie bald die gesamte Menüstruktur auf

einen Blick erfassen. **Hinweis**: Klicken Sie die rechte Maustaste an, um das Menü, das Sie gerade bearbeiten, zu überprüfen. Auf diese Art können Sie die Anzeige mit der Menüzeile, die sie darstellt, direkt vergleichen.

Ist Ihr Menü sehr groß, so möchten Sie vielleicht auch das Editierfenster vergrößern. In diesem Fall können Sie mit den Pfeil-Icons links auf dem Bildschirm durch die gesamte Menüdefinition scrollen.

Das Editor-Menü

Das Editor-Menü enthält eine Reihe von Befehlen, die es Ihnen ermöglichen, die physische Struktur Ihres Menüs zu definieren. Alle Optionen wirken auf den aktuellen Menüpunkt. Er kann direkt aus dem Editierfenster mit der Maus angewählt werden, und wird dann durch invertierte Darstellung hervorgehoben. Das Menü ist in die folgenden drei Hauptbereiche aufgeteilt:

Das Statusmenü für Menüpunkte

Durch diese Optionen wird der Status des gewählten Menüpunktes verändert. In Kapitel 16 des AMOS Handbuchs finden Sie eine ausführliche Darstellung der verschiedenen Menüfunktionen. Beachten Sie, daß die Optionen Tline/Bar/Line und Br.Move/Br.Sta. auf einen ganzen Zweig Ihres Menüs und nicht nur auf einen einzelnen Menüpunkt wirken!

Active/Inactive — Bestimmt, ob der aktuelle Menü- punkt aktiv sein soll oder nicht. Nur aktive Objekte können aus dem fertigen Menü ausgewählt werden.

Line/Bar/Tline — Ordnet die aktuelle Menüstufe in Form einer Zeile, eines Balkens oder einer ganzen Zeile (TOTAL LINE) an.

Linked/Separ. — Hier können Sie bestimmen, ob jeder Punkt in Ihrem Menü mit dem darüberstehenden verbunden werden soll. Eine vollständige Erklärung dieses Konzeptes finden Sie unter dem AMOS-Befehl MENU LINKED.

Br.sta/Br.mov — Abkürzung für Branch Stationary (Zweig fest) und Branch movable (Zweig beweglich). Die Wirkung dieses Befehls ist identisch zu den AMOS Basic-Befehlen MENU MOVABLE und MENU STATIC. So wird bestimmt, ob der aktuelle Menüzweig vom Anwender bewegt werden kann.

It.sta/It.mov — Item stationary/Item movable (Menüpunkt fest/beweglich). Ermöglicht es Ihnen, die Menüpunkte einzeln zu verschieben. Weiter Einzelheiten finden Sie unter den Basic-Anweisungen MENU ITEM MOVABLE/STATIC.

Das Baum-Editiermenü

Durch diese Icons können Sie den Menübaum erweitern und Ihrem aktuellen Menü weitere Punkte hinzufügen. Alle normalen Editieroptionen sind verfügbar, einschließlich Add, Insert und Delete.

Nachdem Sie ein Menü erzeugt haben, können Sie es testen, indem Sie die rechte Maustaste wie üblich drücken. Sind die Menüpunkte beweglich, so können Sie sie beliebig mit Hilfe der Maus auf dem Bildschirm verschieben. Die neuen Positionen werden beibehalten, bis Sie einen Punkt des aktuellen Zweiges löschen oder den Befehl Reset Menu Position einsetzen. In diesem Fall wird das Menü vollständig neu initialisiert.

Add item — Fügt am Ende des aktuellen Menüs einen neuen Menüpunkt hinzu. Als Default wird dem Menüpunkt automatisch die Zeichenkette EMPTY zugewiesen. Dies kann durch eine der zahlreichen Optionen aus dem Zeichen-Menü verändert werden.

Ins item — Fügt an der aktuellen Cursor-Position einen neuen Menüpunkt hinzu.

Branch — Erzeugt am vorhandenen Punkt einen neuen Menüzweig und fügt an diesen Zweig einen neuen Menüpunkt an.

Delete — Löscht einen Menüpunkt aus dem Menü.

Reset menu position — Setzt die Koordinaten aller Menüpunkte, die zum aktuellen Menüzweig gehören, auf ihre Defaultwerte zurück.

Das Zeichen-Menü

Das Zeichen-Menü stellt das Tor zu einem leistungsstarken Zeichensystem auf dem Bildschirm dar, durch das Sie die einzelnen Punkte in Ihrem Menü erzeugen können. Es gibt vier mögliche Menüoptionen:

Normal — Bestimmt das normale Erscheinungsbild eines Punkte in Ihrem Menü.

Highlighted — Definiert die Form Ihres Menüpunktes, nachdem er mit der Maus angewählt wurde.

Inactive — Zeichnet das Objekt, das angezeigt wird, wenn der Menüpunkt durch den AMOS-Befehl MENU INACTIVE deaktiviert wurde.

Background — Bestimmt den Hintergrund für Ihren Menüpunkt. Dieser Bereich wird stets hinter dem aktuellen Menüpunkt dargestellt.

Der Zeichenschirm für Menüpunkte

Dieser Schirm stellt das Kernstück des Menü-Editors dar, denn er erlaubt Ihnen, das genaue Erscheinungsbild jedes einzelnen Menüpunktes auf dem Bildschirm zu definieren.

Alle Menüpunkte bestehen aus eine Reihe von bis zu 21 verschiedenen Grafikelementen. Diese Elemente entsprechen den eingebetteten Anweisungen in einer normalen AMOS Menü-Zeichenkette. Jedes Element kann dabei völlig eigenständig bearbeitet werden, somit ist es ganz leicht, Ihre Menüs nach der Erstellung zu modifizieren.

Angenommen, Sie möchten einen Menüpunkt erzeugen, der einfachen Text enthält. In diesem Fall wird ein leeres Rechteck als erstes Element ausgewählt, und das zweite Element wird dann Ihrem Menütext zugewiesen.

Das Element-Menü

+/-	Über diese Icons können Sie ein Element wählen, das auf dem Bildschirm definiert werden soll. Eine Abbildung dieses Elements können Sie im Fenster Aktuelles Element sehen.
Ins	Fügt an der aktuellen Position ein neues Grafikelement ein. Das 20. Element wird nach hinten weggeschoben und geht daher unwiederbringlich verloren!
Del	Löscht das aktuelle Element.
Push	Speichert ein Element in einem temporären Speicherbereich.
Past	Kopiert ein Bild aus dem Speicher in das aktuelle Element. Diese Daten müssen zuvor durch den Befehl PUSH gespeichert worden sein.

Der Arbeitsschirm

Alle Zeichenoperationen finden in einem eigenen Arbeitsschirm statt. Er enthält ein Bild des fertigen Menüpunktes, so wie er dann schließlich im vollendeten Menü erscheinen wird. Sie können sich Ihr neues Menü jederzeit ansehen, indem Sie die Maus auf die Zeichen-Icons stellen und die rechte Maustaste drücken.

Ein Element zeichnen

Zuerst müssen Sie die Farbe für Ihr Element im Palettenfenster auswählen. Das Icon 1 stellt die Zeichenfarbe und das Icon 2 die aktuelle Hintergrundfarbe dar.

Die Zeichenfarbe kann verändert werden, indem man die Maus auf die obere Hälfte der Palette schiebt und die neue Farbe mit der linken Maustaste anklickt. Entsprechend kann die Hintergrundfarbe durch Auswahl einer neuen Schattierung im unteren Bereich des Palettenfensters angepaßt werden. Für die Umrandung wird automatisch die aktuelle Zeichenfarbe eingesetzt.

Nachdem Sie nun die Farben bestimmt haben, können Sie mit Hilfe der leistungsstarken Funktionen im Zeichen-Menü Ihr Element zeichnen.

Der Zeichenvorgang ist ganz einfach. Stellen Sie den Maus-Cursor in das Fenster mit dem Aktuellen Menüpunkt und halten Sie die linke Maustaste gedrückt. Nun können Sie durch Ziehen der Maus die Form des gewünschten Objektes angeben. Wenn Sie die Maustaste loslassen, wird das Objekt, das Sie erzeugt haben, automatisch dem aktuellen Pinsel zugewiesen. Es kann jetzt durch einmaliges Anklicken mit der linken Maustaste am richtigen Platz eingesetzt werden.

Sollten Sie einen Fehler gemacht haben, so können Sie die aktuelle Einstellung des Pinsels durch Drücken der rechten Maustaste löschen, wenn der Zeiger auf dem Arbeitsbereich steht.

Dabei ist wichtig, daß jedes Element in Ihrem Menüpunkt nur jeweils einem Objekt aus dem Zeichen-Menü zugewiesen werden kann. Vergessen Sie also nicht, die Elementnummer zu erhöhen, bevor Sie Ihrem Design etwas hinzufügen, sonst wird Ihr vorhandenes Element vollständig ersetzt.

Line — Zieht eine gerade Linie in der aktuellen Zeichenfarbe.

Box — Stellt das aktuelle Grafikelement in ein leeres Rechteck.

Bar — Erzeugt einen Balken, der mit dem gegenwärtigen Füllmuster ausgefüllt ist. Das Muster kann über die -/+ Tasten im Settings-Menü geändert werden.

Ellipse — Erzeugt einen leeren Kreis oder eine Ellipse.

Icon — Setzt ein Icon in den Menüpunkt ein. Die Bildnummer kann durch die linken und rechten Pfeiltasten gewählt werden. Um diese Funktion nutzen zu können, muß zuvor die Icon-Bank über die Option Load Memory Bank aus dem Misc-Menü geladen werden.

Bob — Identisch zu Icon, das Bild wird jedoch direkt der Sprite-Bank entnommen.

Text — Über die Tastatur wird Text eingegeben und dem aktuellen Menüelement zugeordnet.

T.Len — Bestimmt die maximale Länge Ihres Menü-Textes.

Make Inverse — Vertauscht die Zeichen- und Hintergrundfarben im aktuellen Element.

Make Border — Faßt das ganze Objekt mit einer sauberen Umrandung ein.

Das Settings-Menü

Durch dieses Menü können Sie den Status der verschiedenen Text- und Grafikoptionen ändern. Die drei n-Icons dienen zur Wahl zwischen den verschiedenen Textmodi wie Fettdruck, Unterstreichen und Kursiv. Durch das Anklicken dieser Icons wird der Text im Fontfenster entsprechend geändert.

-/+ — Auswahl eines Füllmuster zum Einsatz mit dem Befehl BAR. Die gegenwärtig zur Verfügung stehenden Füllmuster werden gleich rechts neben dem Fontfenster angezeigt. Wenn Sie zuvor die Sprite- Bank geladen haben, klicken Sie das -Icon an, um dem aktuellen Füllmuster ein Sprite-Bild zuzuordnen.

Outline — Fügt Ihren Füllmustern einen Rahmen hinzu.

S.Font — Lädt eine Reihe von verfügbaren Fonts von der Programmdiskette, und ermöglicht Ihnen die Auswahl einer neuen Schriftart für Ihren Menütext.

Das Misc-Menü

Push Itm — Speichert einen vollständigen Menüpunkt in einem temporären Speicherbereich. Auf diese Art können Sie Ihre Lieblingsobjekte mehrfach kopieren.

Paste Itm — Ersetzt den aktuellen Menüpunkt durch den zuvor durch PUSH ITM gespeicherten.

Load a memory bank — Lädt eine Gruppe von Icons oder BOBs von der Diskette. Nachdem die Bank installiert ist, wird die neue Palette automatisch in den Hintergrundschirm kopiert.

Der AMAL-Editor

AMAL ist eine erstaunliche neue Animationssprache, die es Ihnen ermöglicht, mühelos die glatten Bewegungen zu generieren, die Sie für ein modernes Spielhallenspiel brauchen, ohne daß Sie auf die komplexe Maschinensprache zurückgreifen müssen.

Die Geschwindigkeit dieses Systems ist phänomenal! AMOS kann leicht bis zu 16 verschiedene AMAL-Animationssequenzen gleichzeitig ausführen, ohne jegliche Auswirkung auf Ihre normalen Basic-Programme. In der Praxis gibt es zwei verschiedene Möglichkeiten, AMAL-Routinen in Ihre Spiele einzubauen. Der übliche Weg besteht darin, alle AMAL-Befehle in einer Zeichenkettenvariable unterzubringen. Dann können Sie Ihre Animationssequenzen mit der folgenden AMAL-Anweisung ausführen lassen:

AMAL n,A$

Das funktioniert bei kleinen Routinen zwar ganz gut, aber die Anwendung bei großen Animationssequenzen, die Sie in einem richtigen Spiel finden können, stellt ein echtes Problem dar. Deshalb haben wir ein nützliches AMAL-Editor- Zusatzprogramm für Sie entwickelt, durch das Sie alle häufig eingesetzten AMAL- Programme in eine einzige Speicherbank packen können. Sie haben nun direkt über die Tastatur Zugriff auf Ihre AMAL-Routinen und können sie mit den normalen Cursor-Tasten auf dem Bildschirm überarbeiten.

Auch das Entfernen von Bugs aus Ihren AMAL-Programmen ist denkbar leicht, denn der Editor verfügt über sein eigenes, eingebautes Überwachungssystem. So können Sie das Programm Anweisung für Anweisung durchgehen. Sie haben direkt vom Schirm Zugang zu allen AMAL-Registern, somit ist es leicht, genau festzustellen, wo ein Fehler aufgetreten ist.

Eine weitere interessante Eigenschaft ist die Möglichkeit, über die Maus Zugang zu komplexen Bewegungsmustern zu haben. Sie können jedem beliebigen Objekt zugewiesen werden und durch eine einzige AMAL PLay-Anweisung wieder abgespielt werden.

Nachdem Sie eine Bank einmal erzeugt haben, können Sie von AMOS-Basic aus auf Ihre neuen Animationssequenzen durch das Aufrufen einer besonderen Version des AMAL-Befehls zugreifen. Das Format dieser Anweisung lautet folgendermaßen:

AMAL n,p

n stellt dabei die Nummer des AMAL-Kanals dar, den Sie zuweisen möchten, und *p* enthält die Nummer eines in der aktuellen AMAL-Speicherbank gespeicherten Programms.

Aber Sie können eigentlich nur wirklich glauben, wie ungemein leistungsstark das AMAL-System ist, nachdem Sie sich mit eigenen Augen davon überzeugt haben. So können Sie zum Beispiel problemlos 90% Ihrer Spielroutinen direkt in AMAL schreiben, und AMOS nur noch dazu einsetzen, langweilige Initialisierungsroutinen und Punktstandtabellen umzusetzen.

Sollte Ihnen etwas ganz Besonderes einfallen, so vergessen Sie bitte nicht, uns eine Kopie davon zu schicken. Wir erwarten Großes von diesem System und würden uns unwahrscheinlich freuen, ein vollständig in AMAL geschriebenes Spiel zu sehen.

Das Laden des AMAL-Editors

Der AMAL-Editor kann entweder als Zusatzprogramm oder als normales Basic-Programm eingesetzt werden. Wenn es als Zusatzprogramm installiert ist, so erfaßt es automatisch eine Kopie der von Ihrem Basic-Programm definierten Speicherbanken.

Der String-Editor

Der String-Editor bietet Ihnen eine Vielzahl von Möglichkeiten, Ihre eigenen AMAL-Programmketten zu erzeugen und Sie in der AMAL Speicherbank zu speichern. Der Bildschirm ist in drei Hauptbereiche aufgeteilt.

Die Statuszeile

Führt den aktuellen Modus auf und zeigt die Nummer des AMAL-Programms an, das Sie gerade bearbeiten. Hier befindet sich auch der Menübalken zur Steuerung des Editors. Zugang zu diesem Menü erfolgt wie üblich über die rechte Maustaste.

Das Auswahlfenster

Wie der Name schon sagt, haben Sie hier die Wahl zwischen den verschiedenen gespeicherten AMAL-Programmen. Um diese Anzeige zu verstehen, müssen Sie wissen, daß alle Nummern von oben nach unten gelesen werden. So lautet der erste Punkt der Liste EE (bedeutet Environment-Editor - siehe weiter unten), der zweite Punkt ist der Kanal 00, der dritte Kanal 01 usw. Das aktive Programm wird stets durch inverse Farben hervorgehoben.

Um eines Ihrer AMAL-Programme zu überarbeiten, klicken Sie nur einfach die entsprechende Kanalnummer mit der linken Maustaste an. Die aktuelle Definition - falls vorhanden - wird dann unverzüglich ins Editierfenster geladen. Alternativ dazu können Sie das Programm auch direkt über die Tastatur auswählen, indem Sie eine der Amiga-Tasten in Verbindung mit der rechten oder linken Pfeiltaste gedrückt halten.

Wenn die Option Synchro Aus aus dem Options-Menü aktiviert wurde, so ist die Anzahl der verfügbaren Kanäle von 16 auf 33 erhöht. Mit dem normalen AMAL Interrupt-System können jedoch nur die ersten 16 dieser Routinen ausgeführt werden.

Das Editierfenster Das Editierfenster funktioniert ähnlich wie der AMOS Programm-Editor, deshalb sollten Sie sich hier eigentlich sofort wie

Zuhause fühlen. Über die normalen Cursor-Tasten können Sie direkt auf jedes AMAL-Programm zugreifen und es editieren. Alle Zeichen werden an der aktuellen Cursor-Position eingegeben. Dieser Cursor kann mit der Maus in Ihrem Programm nach Belieben bewegt werden.

Darüberhinaus gibt es einen umfassenden Satz an Befehlen zum Ausschneiden und Einsetzen im Block- Menü. So ist es leicht, mehreren verschiedenen Objekten die gleiche Animationssequenz zuzuweisen.

Hier sind die verfügbaren Befehle aufgeführt:

Enter Fügt eine Zeile ein

Ctrl+Y Löscht eine Zeile

Tab Springt sofort zum nächsten Tabulator

Cursortasten Bewegt den Cursor um eine Stelle in die entsprechende Richtung.

Shift+Cursor Stellt Cursor auf Anfang/Ende der Zeile oder ganz oben/unten auf den Bildschirm.

Das Verschieben des Editier-Schirms

Stört der Editier-Schirm Ihre gegenwärtige Programmanzeige, so können Sie ihn mit der Maus beliebig verschieben. Stellen Sie den Maus-Cursor auf einen der schwarzen Ziehbalken, die den Schirm umgeben, und halten Sie die linke Maustaste gedrückt. Nun können Sie den Schirm nach unten oder oben schieben. Beachten Sie dabei bitte, daß die Editor-Anzeige stets dem Schirm mit der Nummer sieben zugewiesen wird. Also setzen Sie diesen Schirm bitte nicht in Ihren Basic- Programmen ein.

Das Aufrufen Ihrer AMAL-Programme

Nachdem Sie Ihre AMAL-Programme gespeichert haben, können Sie sie über eine Reihe von Optionen aus dem Editier-Menü ablaufen lassen. Im folgenden sind die wichtigeren Funktionen kurz aufgeführt.

Start alle Ruft die AMAL-Environmentkette (siehe weiter unten) auf und führt gleichzeitig Ihre existierenden AMAL-Programme aus.

Start aktuell Initialisiert den Schirm und führt das AMAL-Programm aus, das Sie gerade überarbeiten.

Debug Steigt in den AMAL-Debugger ein.

Die Umgebungsbefehle

In der Praxis laufen wenige AMAL-Programme völlig isoliert ab. Die große Mehrheit der AMAL-Routinen müssen erst sorgsam in AMOS-Basic initialisiert werden, bevor Sie eingesetzt werden können.

Deshalb umfaßt der AMAL-Editor eine verkürzte Version des AMOS-Basic Interpreters, den man als Environment-Generator (Environment = Umgebung) bezeichnet. Auf diese Art können Sie all Ihre Initialisierungsroutinen direkt vom AMAL-Editor aus durchführen, ohne ständig in AMOS-Basic zurückgehen zu müssen.

Diese Befehle entsprechen ihren Gegenstücken in AMOS-Basic zwar im wesentlichen, wir mußten jedoch manche der Anweisungen vereinfachen, um Speicher zu sparen. Hier ist eine vollständige Liste der von diesem System unterstützen Befehle:

SPRITE OFF	Entfernt alle Sprites vom Schirm
BOB OFF	Entfernt BOBs
RAINBOW DEL	Löscht alle Regenbogeneffekte
	REMark, Bemerkung am Anfang der Zeile
SCREEN OPEN	Wie bei AMOS
SCREEN DISPLAY	Wie bei AMOS
SCREEN OFFSET	Wie bei AMOS
SCREEN	Wie bei AMOS
SCREEN CLOSE	Schließt alle Schirme
DOUBLE BUFFER	Wie bei AMOS
DUAL PLAYFIELD	Wie bei AMOS
DUAL PRIORITY	Wie bei AMOS
LOAD IFF "Name",Schirm	Sie müssen eine Schirmnummer angeben!
COLOUR	Wie bei AMOS
GET SPRITE PALETTE mask	Sie müssen die Maske angeben
FLASH	Wie bei AMOS
FLASH OFF	Wie bei AMOS
SET RAINBOW	Wie bei AMOS
RAINBOW	Wie bei AMOS
LOAD "Name",Bank	Sie müssen Bank angeben
ERASE Bank	Wie bei AMOS
BOB	Wie bei AMOS
SET BOB	Wie bei AMOS
SPRITE	Wie bei AMOS
SET SPRITE BUFFER	Wie bei AMOS
HIDE ON	Wie bei AMOS
UPDATE EVERY	Wie bei AMOS
CHANNEL TO SPRITE Kanal,Sprite	Hier anders!
CHANNEL TO BOB Kanal,Bob	
CHANNEL TO SCREEN DISPLAY Kanal,Schirm	

CHANNEL TO SCREEN OFFSET Kanal,Schirm
CHANNEL TO SCREEN SIZE Kanal,Schirm
CHANNEL TO RAINBOW Kanal,Schirm
SET REG Nummer,Wert Stellt AMAL-Register A-Z (0-25) ein

Außerdem gibt es eine Reihe von nützlichen Abfrageanweisungen. Die Syntax dieser Befehle lautet im allgemeinen folgendermaßen:

BEFEHL Bedingung : Anweisungen

Die auf die Bedingung folgenden Anweisungen werden nur ausgeführt, wenn die Bedingung RICHTIG ist, also zutrifft. Ist die Bedingung FALSCH, so werden diese Anweisungen völlig ignoriert, und das Programm springt sofort zur nächsten Zeile Ihrer Environment-Routine.

IF SCREEN Nummer (Richtig, wenn Schirm GEÖFFNET wurde)
IF NOT SCREEN Nummer (Richtig, wenn Schirm derzeit GESCHLOSSEN ist)
IF BANK Nummer (Richtig, wenn Bank reserviert wurde)
IF NOT BANK Nummer (Richtig, wenn Bank nicht reserviert wurde)
IF REG Nummer,Wert (Richtig, wenn Register A-Z (0-25) gleich Wert)
IF NOT REG Nummer,Wert...

Sie erhalten über den AMAL-Editor Zugang zum Environment-Programm, ganz genauso, als handle es sich hier um ein normales AMAL-Programm. Klicken Sie einfach nur die Option EE im Auswahlfenster an tippen Sie Ihre Prozedur über die Tastatur ein. Ihr neues Programm wird automatisch als Teil der aktuellen Speicherbank gespeichert. Jedesmal, wenn Sie ein AMAL-Programm vom Editor aus ablaufen lassen, wird es ausgeführt.

Achtung! Wenn Sie Ihre AMAL-Routinen von einem Basic-Programm aus ablaufen lassen möchten, so müssen Sie Ihre Environment-Befehle erst in AMOS Basic-Anweisungen umsetzen, und dann die entsprechenden Zeilen am Anfang Ihres Codes hinzufügen. Normalerweise ist das aber ganz problemlos.

Anmerkung: Wenn Sie BOBs einsetzen, so MÜSSEN Sie eine SCREEN-Anweisung haben, um die Anzeige auf dem Bildschirm zu steuern, sonst werden die BOBs auf den Editor-Schirm (Schirm Nummer 7) gezeichnet.

Der Bewegungs-Editor

Mit dem Play-Editor können Sie komplizierte Bewegungsmuster über den Bildschirm eingeben. Dabei wird einfach wiederholt die Position des Maus-Cursors abgefragt, während er sich über den Bildschirm bewegt und diese Daten in die AMAL Speicherbank gestellt. Nachdem Sie dieses Muster einmal erzeugt haben, können Sie es dazu einsetzen, so ziemlich jedes beliebiges Objekt zu animieren.

Zugang zu dieser Funktion erhalten Sie über die Play-Option im Editiermenü. Wie beim String-Editor kann auch hier die Anzeige beliebig auf dem Bildschirm positioniert werden.

Das Aufzeichnen eines Bewegungsmusters

Das Aufzeichnen eines Bewegungsmusters ist einfach. Zuerst müssen Sie die Nummer des Musters, das Sie definieren möchten, durch Hervorheben des entsprechenden Menüpunktes im Auswahlfenster bestimmen.

Nun rufen Sie die Option Record aus dem Bewegungs-Menü auf. Sie werden sofort aufgefordert, die aktuelle Aufzeichnungsgeschwindigkeit einzugeben. Sie bestimmt die Verzögerung in 50stel Sekunden, die zwischen jeder weiteren Abfrage in Ihrem neuen Bewegungsmuster liegt.

Es besteht eine Wechselwirkung zwischen der Aufzeichnungsgeschwindigkeit und dem in der AMAL-Bank eingenommenen Speicherplatz. Die glattesten Animationseffekte erfordern eine hohe Abfragehäufigkeit, und werden am besten mit Hilfe der Defaultparameter erzeugt. Wenn Sie jedoch beabsichtigen, eine große Anzahl von Mustern gleichzeitig zu generieren, so können Sie das möglicherweise auch mit einer etwas niedrigeren Einstellung zu Wege bringen. Auf diese Art können Sie eine beträchtliche Menge von wertvollen Speicherplatz sparen.

Nachdem Sie die Geschwindigkeit eingegeben haben, können Sie nun mit der Aufzeichnung beginnen. Stellen Sie die Maus auf den Anfangspunkt Ihrer Animationssequenz und drücken Sie ENTER. Nun werden Ihre Mausbewegungen ständig abgefragt, bis Sie diesen Modus durch Drücken der linken Maustaste verlassen.

Seien Sie bitte nicht enttäuscht, wenn Sie nicht von Anfang an perfekte Ergebnisse erhalten. Es kann ein Weilchen dauern, bis Sie den Dreh heraushaben, und Sie müssen vielleicht etwas experimentieren, bis Sie die bestmöglichen Ergebnisse erhalten.

Das Überprüfen Ihrer Bewegungsmuster

Wenn Sie eine Bewegungsfolge definiert haben, können Sie sie durch die Funktion Play back im Play-Editor wieder abspielen.

Alternativ dazu möchten Sie aber vielleicht Ihre Muster zusammen mit den Objekten, die Sie in Ihrem Spiel tatsächlich animieren werden, überprüfen. Dazu gehen Sie in den String-Editor zurück und geben zum Beispiel folgende Anweisung ein:

```
PL n
```

Dabei stellt *n* die Nummer der Bewegungsfolge dar, die abgespielt werden soll.

Vergessen Sie aber nicht, Ihre Objekte mit dem Environment-Programm zu initialisieren, bevor Sie diesen Befehl einsetzen, sonst wird die PL-Anweisung nämlich völlig ignoriert.

Zusammenfassung der Play-Befehle

Edit-Menü

Zurück zum Haupt - Editor	Springt zum Haupteditor zurück

Das Bewegungsmenü

Aufnahme Zeichnet ein Bewegungsmuster auf und speichert es in der AMAL-Bank

Abspielen Spielt eine bereits aufgezeichnete Sequenz unter Einsatz des Maus-Cursors ab.

Einfügen Fügt ein neues Muster ein und beginnt dabei an der aktuellen Cursor- Position. Alle Muster nach diesem Punkt werden um eine Stelle nach rechts gerückt. Achtung! Das 48. Muster in dieser Reihe geht verloren!

Entfernen Löscht den aktuellen Bewegungspfad. Alle übrigen Muster werden eine Stelle nach links gerückt.

Der AMAL-Monitor

Der AMAL-Monitor enthält eine Reihe von leistungsstarken Funktionen, die das Entfernen von Bugs in völlig ungeahntem Maß erleichtern. Zugang erhalten Sie, indem Sie Debug im Edit-Hauptmenü wählen.

Zu Anfang sehen Sie eine komplex erscheinende Anzeige. Oben auf diesem Schirm befindet sich ein Auswahlfenster, das eine vollständige Liste aller derzeit aktiven Programme enthält. Als Default werden alle AMAL-Programme im Speicher gleichzeitig ausgeführt. Wenn Sie also nur eine bestimmte Auswahl dieser Programme ablaufen lassen möchten, so müssen Sie die nicht erwünschten Routinen manuell deaktivieren. Vergessen Sie dabei aber bitte nicht, daß aktive Programme durch inverse Farben hervorgehoben werden.

Ändern eines Registers

Externe Register: Links vom Monitorfenster befindet sich eine Liste aller 26 externen Register. Sie können jederzeit einfach durch Anklicken des entsprechenden Registers und Eingeben eines neuen Werts im Hexadezimalformat verändert werden.

Interne Register: Jedes AMAL-Programm verfügt über seinen eigenen Satz an internen Registern. Sie werden in einem kleinen Rechteck in der Mitte des Monitorfensters angezeigt. Das Format dieser Register lautet folgendermaßen:

R(Kanalnummer,Registernummer)

Sie können die internen Register jedes AMAL-Programms untersuchen, indem Sie eines der Pfeil-Icons unten an der Registeranzeige anklicken. Um ein Register einzustellen, wählen Sie es mit der linken Maustaste an und geben einfach einen neuen Wert über die Tastatur ein (verwenden Sie dazu das Hexadezimalformat).

Das Entfernen von Bugs aus Ihren Programmen

Die Hauptoptionen hierzu werden über eine Gruppe von fünf Tasten rechts von der Anzeige gesteuert. Alle Befehle können entweder durch Auswahl eines Icons mit der Maus oder durch Drücken der entsprechenden Taste ausgeführt werden.

Bevor Sie diese Optionen jedoch einsetzen können, müssen Sie das AMAL-System erst durch Anklicken der Init-Option oder Drücken der I-Taste auf der Tastatur initialisieren. Dies läßt eine Environment-Routine ablaufen und bereitet den Monitor für den Einsatz vor. Sie können nun Ihr Programm mit den folgenden Befehlen überprüfen:

R (Run) — Läßt die ausgewählten AMAL-Programme ablaufen, bis eine Taste gedrückt wird.

G (Go until) — Führt Ihre AMAL-Programme aus und kehrt zum Monitor zurück wenn das gewählte Register einen bestimmten Wert enthält. Pausen im Ablauf können erzeugt werden, indem man einem Register an den entscheidenden Punkten im AMAL-Programm einen bestimmten Wert zuweist. Zum Beispiel:

```
Let RZ=1
```

Anmerkung: — Wenn Sie die Hexadezimalform einsetzen, vergessen Sie nicht, am Anfang Ihres Abfragewertes ein $- Zeichen anzufügen.

S (Single Step) — Führt einen Einzelschritt Ihres AMAL-Programms aus. Funktioniert auch mit dem AMAL-Play-Befehl.

(Esc) — Kehrt in das Haupteditierfenster zurück.

Anmerkungen: Sollte das Monitorfenster bei Ihrer Programmanzeige störend wirken, so können Sie es mit der Maus verschieben. Bewegen Sie es einfach anhand der Bewegungsbalken unten und oben auf dem Bildschirm. Alle Hardware-Sprites werden vor dem Monitorfenster angezeigt.

Die Anweisungen zum ASCII-Leseprogramm

AMOS wird ständig weiterentwickelt, und es werden stets neue Funktionen hinzugefügt. Diese Funktionen sind in einer Reihe von ASCII-Dateien auf der Programmdiskette dokumentiert. Um Ihnen das Leben zu erleichtern, haben wir in dieses Paket auch ein leistungsstarkes ASCII-Leseprogramm integriert. So können Sie die gesamten Dokumentationsdateien direkt aus AMOS-Basic heraus lesen (und ausdrucken). Durch

Drücken der HELP-Taste kann das ASCII-Leseprogramm von der AMOS-Programmdiskette geladen werden. Hoffentlich lesen Sie diese Datei mit Hilfe des Leseprogramms. Dann haben Sie nämlich schon die Funktionen einiger der Schirm-Icons herausgefunden. Hier finden Sie nun eine ausführliche Erklärung der verschiedenen Befehle:

Das AMOS-Logo

Aussteigen aus dem ASCII-Lese-Zusatzprogramm und Rückkehr zum AMOS Editor-Schirm.

Das TV/Monitor-Icon

Als Default zeigt das ASCII-Leseprogramm stets alle Dokumentationsdateien im europäischen PAL-Bildschirmformat an. Sind Sie ein amerikanischer Anwender und verfügen Sie über eine NTSC-Anzeige, so können Sie dieses Icon anklicken, um das Textfenster zu verkleinern, so daß es vollständig auf den Bildschirm paßt. Der Buchstabe "N" erscheint dann im Monitor-Icon, um die neue Einstellung hervorzuheben.

Das Disketten-Icon

Dies ist bei weitem das wichtigste Icon. Es lädt eine Textdatei von der Diskette und zeigt sie auf dem Amiga-Bildschirm an.

Das Ausdruck-Icon

Wenn Sie über einen Drucker verfügen, so können Sie über dieses Icon jeden beliebigen Teil dieses Dokuments ausdrucken lassen. Achtung! Um aus diesem Programm heraus ein Dokument ausdrucken zu können, benötigen Sie mindestens 1 MByte Speicherkapazität. Das hat aber eigentlich nichts mit AMOS zutun. Es liegt an der enormen Größe des Amiga "PRINTER"-Devicetreibers. Wenn Sie jedenfalls versuchen, über eine Maschine zu drucken, bei der nur ein Speicher von 512 KByte installiert ist, so erscheint das Warnrechteck (Alert) und teilt Ihnen mit, daß für das Drucken nicht genug Speicher zur Verfügung steht.

Die Pfeil-Icons Oben/Unten

Diese beiden Icons kommen erst ins Spiel nachdem Sie eine Textdatei geladen haben. Sie ermöglichen es Ihnen, nun im Dokument nach oben und unten zu gehen.

Pfeil nach OBEN: Zeigt die vorhergehende Seite an.
Pfeil nach UNTEN: Geht auf die nächste Seite des Dokuments.

Das Link-Icon

Dieses Anzeige-Icon erscheint ganz oben in der rechten Ecke des Bildschirms. Damit kann man ein Hilfe-Menü erzeugen, das automatisch erscheint, wenn Sie eine Datei mit

der Erweiterung ".Lnk" laden. (Sehen Sie auf der Programmdiskette nach, welche Link-Dateien verfügbar sind). Wenn Sie eine der Menüoptionen anklicken, so wird die Textdatei, die dieser Option zugewiesen wurde, sofort durch das ASCII-Leseprogramm geladen. Sie können dann das Dokument durchgehen, als hätten Sie es selbst aufgerufen. Das Menü kann jederzeit einfach durch Anklicken des Anzeige-Icons wieder aufgerufen werden.

Anmerkung: Wenn Sie ein Link-Menü laden, müssen Sie das Verzeichnis stets mit der Option SETDIR aus dem Datei-Requester dem ROOT-Verzeichnis Ihrer aktuellen Diskette zuweisen. Sollten Sie das vergessen, so wird jedesmal, wenn Sie versuchen, eine Menüoption auszuwählen, ein Diskettenfehler gemeldet.

Wenn Sie Ihre eigenen Hilfe-Menüs erstellen möchten, so können Sie dazu den leistungsstarken Link-File-Creator auf der AMOS-Programmdiskette einsetzen. Es ist dann ganz einfach, für jede Ihrer AMOS Basic-Applikationen Hilfe-Menüs zu erzeugen.

Das Lesen großer Dateien

AMOS Anwender, die über ein Megabyte oder mehr Speicherkapazität verfügen, können die Größe der Dateien, die das ASCII-Leseprogramm bearbeiten kann, mit dem Befehl SET BUFFER erweitern. Der Variablenpuffer sollte auf:

(Maximale Dateigröße*2)+1024 KByte

vergrößert werden.

Das Link-File-Creator-Zusatzprogramm

Mit dem AMOS Link-File-Creator können Sie kleine Menüs für den Einsatz mit dem ASCII-Leseprogramm erzeugen. Dazu müssen Sie nur einfach eine Reihe von Menütiteln eingeben und Sie mit einer Textdatei verbinden.

Anstatt alle Menüoptionen aufzuführen, gehen wir mit Ihnen nun Schritt für Schritt das Erstellen eines einfachen Menüs durch.

1 Schritt: Wählen Sie die Option "CREATE LINK FILE" aus dem Menü. Das Programm fragt Sie nun nach dem Titel Ihres neuen Menüs. Tippen Sie "DAS IST EIN TESTMENUE" ein und drücken Sie ENTER.

2 Schritt: Sie sehen jetzt ein hübsches Layoutformular auf dem Bildschirm. Am Anfang steht der Cursor in dem Rechteck "OPTION TITLE". Geben Sie hier "DAS IST OPTION 1" ein und drücken Sie ENTER. Nun erscheint ein Datei-Requester auf dem Bildschirm, und Sie werden aufgefordert, den Namen der Link-Datei auszuwählen. Hier wählen Sie die Datei, die geladen werden soll, wenn die Option aus dem ASCII-Leseprogramm heraus gewählt wird. Für unser Beispiel können Sie nun jede beliebige Datei wählen.

3 Schritt: Und nun sollten Sie über ein schönes Menü mit einem Titel, einer eingegebenen Option und weiteren sieben leeren Optionen verfügen. Jede Option kann einfach durch Anklicken bearbeitet werden. Dann erhalten Sie jeweils den Schirm mit "OPTION TITLE", den wir oben beschrieben haben. Mit diesem Vorgehen können Sie alle Optionen in Ihrem Menü definieren. Ist das nicht kinderleicht? Auf dieselbe Art können Sie auch den Menütitel verändern. Wählen Sie nur einfach MENU TITLE mit der Maus an und geben Sie Ihren neuen Titel über die Tastatur ein.

4 Schritt: Nachdem Sie Ihr Menü erstellt haben, müssen Sie es auf Diskette speichern. Rufen Sie die Option SAVE oder SAVE AS aus dem EDIT-Menü auf. Nun wird ein normaler Datei-Requester angezeigt. Sie können Ihren Dateinamen jetzt wie üblich eingeben. Vergessen Sie aber nicht, die Dateierweiterung .lnk am Ende mitanzugeben.

Tips und Tricks

In erster Linie wurde der Link-File-Creator dafür konzipiert, eine buchähnliche Struktur für eine Reihe kürzerer Textdateien zu bieten. Lange Textdateien verlangsamen die Arbeitsgeschwindigkeit des ASCII-Leseprogramms und schläfern auch den potentiellen Leser ein! Zweitens lassen Sie am besten die Finger vom Link-File-Creator, bis Sie tatsächlich all Ihre Textdateien über das Textverarbeitungsprogramm eingegeben haben. Dann fällt es Ihnen nämlich sicher wesentlich leichter, gute Menü-Überschriften usw. zu finden.

Wenn Sie etwas risikofreudig sind, möchten Sie den Link Editor vielleicht selbst erweitern. Vielleicht können Sie die Menüs mit ein paar interessanten Grafiken aufpeppen? Oder wie wäre es mit der Funktion "DELETE OPTION"? Wenn Sie ein paar gute Veränderungen durchgeführt haben, vergessen Sie bitte nicht, Sie an Mandarin Software zu senden. Wer weiß, vielleicht sind sie sogar so gut, daß wir sie in die nächste AMOS-Version miteinbeziehen können.

Tastaturdefinitionswerkzeug

AMOS enthält ein hochentwickeltes Werkzeug zum Definieren der Tastatur. So können Sie das Layout Ihrer gesamten Tastatur einfach durch Anklicken mit der Maustaste verändern. Das Einzige, was Sie während dieses Programms über die Tastatur eingeben müssen, ist ein Dateiname zum Speichern oder Laden einer Tastaturdefinition.

Wenn Sie das Programm aufrufen, erscheint eine Darstellung Ihrer Amiga-Tastatur. In manchen Ländern wird den Anwendern dabei auffallen, daß einige zusätzliche Tasten auf diesem Bild sind. Dafür gibt es einen einfachen Grund: manche Amiga- Tastaturen haben zusätzliche Tasten.

All diese Tasten können nun neu definiert werden, um besondere Anforderungen zu erfüllen, abgesehen von der ENTER-, der ESCAPE- und der BACKSPACE- (Rück-Taste).

Nun wollen wir uns das an einem Beispiel ansehen und eine neue Tastaturdefinition erstellen.

1 Schritt: Klicken Sie die Alt-Taste auf dem Bildschirm mit der linken Maustaste an. Nun wird die Tastatur neu gezeichnet und die Alt-Taste hervorgehoben. Sie fragen sich vielleicht, warum einige der Tasten mit einem kleinen Viereck umrahmt sind. Dieses Viereck zeigt an, daß diese Taste über keinen echten ASCII-Wert verfügt (die Taste hat keinen anzeigbaren Wert).

2 Schritt: Klicken Sie nun das Viereck an, das normalerweise die Taste 1 (oben links auf der Tastatur) enthält. Nun wird in dem langen, schmalen Rechteck oben auf dem Bildschirm die folgende Information angezeigt:

KEY SCANCODE (Tasten-Scancode): Enthält den Scancode der hervorgehobenen Taste.

DEFAULT VALUE (Defaultwert): Dieser Wert wird der Tastaturdefinition entnommen, die automatisch geladen wird, wenn das Programm das erste Mal abläuft.

CURRENT VALUE (Aktueller Wert): Gibt den ASCII-Wert an, der stets ausgewiesen wird, wenn man diese Taste drückt.

3 Schritt: Nun können wir das Zeichen auswählen, das wir der ALT-1-Taste zuweisen möchten. Dazu klickt man die Pfeiltasten an, die sich direkt über der Tastatur befinden. Gehen Sie mit der Maus auf die rechte Pfeiltaste und halten Sie die linke Maustaste gedrückt, bis in dem Viereck für die Zeichen ein großes A erscheint. (ASCII-Wert 65).

Diesen Wert können Sie nun einfach durch Anklicken des Zeichen-Vierecks zuweisen (Es ertönt außerdem ein akustisches Signal).

4 Schritt: Zum Schluß können Sie Ihre Zuweisung überprüfen, indem Sie die Maus in die Statuszeile oben auf dem Bildschirm stellen. Diese Zeile enthält alle Informationen über die Tastendefinition. (Sie beginnt mit KEY SCANCODE:).

Wenn Sie dieses Icon mit der linken Maustaste anklicken, so erscheint ein kleiner schwarzer Cursor in dem Rechteck. Sie können Ihre neue Taste jetzt durch Drücken der Tastenkombination "Alt" + "1" überprüfen. Es erscheint nun hoffentlich ein großes "A".

Die Menü-Optionen

Man kann jederzeit eine Menüzeile aufrufen, indem man die rechte Maustaste gedrückt hält. Das EDIT-Menü enthält eine Reihe von einfachen Befehlen, die Ihnen das Laden und Speichern verschiedener Tastaturlayouts auf Diskette ermöglichen. Dieses Menü unterstützt auch die Optionen RESTORE KEYS und RESTORE KEYBOARD, durch die Sie eine oder mehrere Tasten auf ihre Defaultwerte zurücksetzen können. Das kann sehr nützlich sein, weil Sie so ganz mühelos eventuelle Fehler korrigieren können.

Wenn Sie außerhalb von Großbritannien wohnen und ein neues Layout erstellt haben, das für Ihre spezielle Tastatur geeignet ist, senden Sie es doch bitte an Mandarin Software. Wir hoffen nämlich, daß wir dann eine Public-Domain-Diskette mit den neuen Layouts für alle Länder auf der Welt herausgeben können (zumindest für alle, die Commodore Amigas haben!).

AMOS Sample-Bank-Zusatz

Mit diesem Programm können Sie Samples in den Speicher laden, so daß Sie eine Speicherbank zum Einsatz in Ihren AMOS-Programmen erstellen können.

Aber warum sollen wir diese Methode überhaupt einsetzen? Warum nicht einfach das normale IFF-Format verwenden? Nun, das IFF-System ist zwar sehr gut, aber in Bezug auf die Speicherverwaltung ist es nicht gerade effizient. Da Samples aber oft besonders groß sind, ist es sinnvoll, den benötigten Speicherplatz auf ein Minimum zu beschränken. Auf diese Art können Sie dann mehr und größere Samples in Ihr AMOS-Programm packen.

Und wie man ihn einsetzt

Der Sample-Bank-Zusatz ist völlig menügesteuert. Wie üblich erscheint das Menü erst, wenn Sie die rechte Maustaste gedrückt halten. Am besten wählen Sie die Töne, die Sie in Ihrer Bank einsetzen möchten aus, bevor Sie dieses Programm aufrufen, dann es besteht keine Möglichkeit, die Samples direkt aus der Utility heraus zu spielen. Das spart Speicher und erlaubt es dem Anwender, bis zu 40 KByte große Samples bei nur 512 KByte Speicherkapazität einzusetzen. ANMERKUNG: Anwender, die über zusätzliche Speicherkapazität verfügen, sollten den Speicher am Anfang des Programms durch den Befehl SET BUFFER erweitern, sonst müssen sie sich mit den normalen 40 KByte begnügen.

Eine Bank können Sie erstellen, indem Sie einfach jedes Sample mit der Option "LADE SAMPLE" aus dem EDIT-Menü in den Speicher laden. Das Programm erkennt sowohl Amiga Sample-Ursprungsdaten (oder sogenannte "DUMPS" - wie sie im Programm Perfect Sounds bezeichnet werden), als auch mit dem STOS Maestro erzeugte Samples. Ist das Sample im Ursprungsformat, so fragt das Programm nach dem Default für die Sample-Frequenz (in KHz).

Wenn Sie dann zum Konvertieren und Speichern der Töne auf Diskette bereit sind, wählen Sie einfach die Option SICHERN aus dem EDIT-Menü. Die Konvertierung kann zwar ein bißchen dauern, aber das ist schon alles!

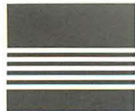
Der Speichereinsatz

Die Speicherkapazität, die dem Programm für das Speichern von Samples zur Verfügung steht, hängt von der aktuellen Größe des Textpuffers ab. Wie ich schon bereits erwähnte, kann diese mit dem Befehl SET BUFFER am Anfang des Programms eingestellt werden. Wenn Sie diesen Wert nun erhöhen, so steht für die Konvertierung mehr Speicherplatz zur Verfügung. Aber Achtung, das Programm muß die Samples und die neue Bank tatsächlich im Speicher halten. Den für das Konvertierprogramm erforderlichen Speicherplatz berechnet man einfach, indem man die Größe der Samples mit zwei multipliziert. Ist dieser Wert geringer als der verfügbare freie Speicher, so

können Sie die Zuweisung problemlos erhöhen. Achtung: Bei Computern mit Speichererweiterung kann zu diesem Zweck nur FAST-Speicher eingesetzt werden.

Vorschläge

Sie könnten zum Beispiel mit dem Programm die Samples ins AMOS-Format konvertieren lassen, während sie geladen werden, oder einige neue Routinen hinzufügen, um Samples von anderen Computern wie dem Acorn Archimedes zu konvertieren. Vergessen Sie bitte nicht, uns Ihre Erweiterungen des Programms zuzusenden, denn sie sind sicher auch für die anderen Anwender von großem Interesse.



27: Spiele

Amosteroids

Im Jahre 2020 ist die Erde verwüstet, von der Umweltverschmutzung und Überindustrialisierung zerstört. Sie haben sich als Freiwilliger dazu gemeldet, ein Super-Raumschiff, das letzte Iconoclast-MKII-Kampfschiff zu steuern. Unglücklicherweise mußten die Wissenschaftler, damit die starken Gravitationsfelder überwunden werden können, auf Kosten der Waffenausrüstung Hochenergie-Antischwerkrafttriebwerke installieren. Ihr einziger Schutz (und gleichzeitig Ihre einzige Angriffsmöglichkeit) ist ein Umkehrschutzfeld, das seine Energie aus Ihrer Umgebung zieht. Je mehr Asteroiden auf Sie zukommen, desto mehr verliert Ihr Raumschiff an Beweglichkeit, und desto stärker wird Ihr Schutzschild. Leider blieb aufgrund des eiligen Einbaus Ihrer neuen Triebwerke (und des Schutzschirms) keine Zeit für eine Notfalltaste. So wird eine Kettenreaktion in den Antischwerkrafttriebwerken ausgelöst, wenn Ihr Schutzschirm keine Energie mehr hat, die schließlich in einer spektakulären Implosion (und damit Ihrem schrecklichen Ende) gipfelt.

Steuerung

JOYSTICK LINKS	Rotiert das Raumschiff nach links
JOYSTICK RECHTS	Rotiert das Raumschiff nach rechts
JOYSTICK UP	Aktiviert die Antischwerkrafttriebwerke
FEUERKNOPF	Aktiviert Schutzfelder
P-TASTE	Pause/Pause beenden

Entwickelt von:

Programmierung und Konzept: Peter J. Hickman
Musik: D.J. Nuttall
Grafik: Peter J. Hickman, C.D. White und Unterstützung von Adam Fothergill.
Wir bedanken uns bei Adam Fothergill für die Tips und Tricks.

Technische Info

Amosteroids konfiguriert sich anhand des verfügbaren RAM-Speichers selbst. Das bedeutet leider, daß Sie auf einem Amiga mit nur 512 KByte keine Musik und keine Hintergrundschirme haben werden (Sie können aber getrennt mit der Musik und den Schirmen spielen). Die Hintergrundschirme werden durch Copper-Regenbogen ersetzt, die fast so gut aussehen wie Bilder.

Allgemeine Info

Amosteroids ist das Ergebnis monatelanger Arbeit mit AMAL (der **AM**OS **A**nimation **L**anguage). Hier werden sehr umfangreiche und komplizierte Zeichenketten eingesetzt, um die hohe Geschwindigkeit von Raumschiff und Asteroiden zu ermöglichen. Alles wird aus AMAL heraus gesteuert, der Basis-Teil des Programms bewirkt nur das Explosionsgeräusch und aktualisiert die Punktstandtabelle. Es interessiert Sie dabei vielleicht auch, daß alle externen AMAL-Register als Mitteilungssystem dienen, die so dem Basic-Teil des Programms melden, wenn ein Asteroid explodiert ist, worauf dann das Explosions-Sample abgespielt wird.

Vorschläge
Vielleicht möchten Sie dieses Programm verbessern. Sie könnten zum Beispiel eine Option hinzufügen, durch die zwei Spieler gleichzeitig spielen können. Sie können auch einige intelligente Aliens hinzufügen, die das Raumschiff angreifen, und eine Art Bonussystem (so kann vielleicht der Schutzschirm wieder aufgeladen werden).

Auf alle Fälle wünschen wir Ihnen viel Spaß, und vergessen Sie nicht, uns Ihre Programmerweiterungen und -verbesserungen zuzuschicken, wir würden Sie uns gerne ansehen!

Castle AMOS
Das Ganze spielt irgendwann in der Zukunft. Sie haben Ihr AMOS-Meisterwerk vollendet, aber irgendwie hapert es noch etwas an der Geschwindigkeit. Also brauchen Sie unbedingt den ganz neu erschienenen AMOS-Compiler.

Sie ziehen also Ihre Stiefel und Ihren Regenmantel an (Pfui, es gießt wieder mal in Strömen!), und eilen zur nächsten Bahnstation, um zu Ihrem Lieblings- Computerladen zu fahren. Da kommen Sie nun am Hauptbahnhof an, laufen die Straße hinunter und was sehen Sie? Nichts, überhaupt nichts, denn der Laden und auch der Bahnhof sind verschwunden (Ihnen entgeht aber wirklich auch nichts!).

Nun dämmert es Ihnen: entweder konnte der Fahrkartenkontrolleur Sie nicht leiden, oder ein Kurzschluß im neuen, automatisierten Drehkreuz hat ein interdimensionales Tor geöffnet und Sie nach Transsilvanien versetzt. Und mit einer Rückfahrkarte bekommen Sie den Compiler in Transsilvanien wohl nicht, aber vielleicht finden Sie des Rätsels Lösung in der Burg dort hinten..

Steuerung
Im Gegensatz zu vielen anderen Abenteuerspielen können Sie bei Castle AMOS den größten Teil des Spiels mit der Maus steuern. Alle Befehle können Sie aus der Scroll-Liste im rechten Scroll-Balken auswählen.

Wenn Sie zu einem neuen Handlungsort gehen, oder den PICTURE-Befehl verwenden, um sich den aktuellen Ort genauer anzusehen, so werden die Scrolls vom Schirm weggezoomt. Durch Anklicken einer der Maustasten werden sie wieder sichtbar. Mehr brauchen Sie eigentlich garnicht zu wissen, sonst machen wir es Ihnen zu leicht!

Entwickelt von:
Programmierung und Konzept: Abdul M. Kalim und Peter J. Hickman
Musik: D.J. Nuttall
Grafik: C.D. White und Peter J. Hickman
Wir bedanken uns bei Richard Vanner (Hallo, Graf Dick!) und François dafür, daß wir sie verstümmeln durften!

Magic Forest
Es wird herbstlich im Zauberwald, und die Früchte fallen von den Bäumen. Willi hat die ehrenvolle Aufgabe, die Äpfel aufzufangen, bevor sie auf den Boden fallen und kaputtgehen. Aber ein paar böse, bleichgesichtige Apfelmanscher sind auf die Sammelplattformen gelangt und werfen die Äpfel herunter!

Sie steuern nun Willi und müssen versuchen, auf jeder Stufe eine bestimmte Quote an Äpfeln zu sammeln. Für rote Äpfel gibts 70 Punkte (ein Apfel wird von Ihrer Quote abgezogen), für rote Kirschen 100 Punkte (zwei Äpfel werden von der Quote abgezogen) und für blaue Äpfel 200 Punkte (drei Äpfel werden abgezogen).

Wenn Sie zuviele Äpfel fallenlassen, einen vergifteten Apfel auffangen, oder einen der Apfelmanscher berühren, verlieren Sie eines Ihrer drei Leben. Die Apfelmanscher singen ständig einen Zauberspruch, der nach kurzer Zeit ein Gewitter hervorruft. Und nun sollten Sie nicht vergessen, daß es äußerst gefährlich ist, während eines Gewitters unter einem Baum zu stehen - außer Sie vertragen Elektroschocks ganz gut! Wenn Sie sich also nicht sputen, so wird Ihnen der Blitz bald Beine machen!

Steuerung

Willi wird über den Joystick in Port 1 (nicht dem Maus-Port) bewegt.
Drückt man den Joystick nach links, geht auch Willi nach links, und Drücken nach rechts bringt Willi nach rechts.
Drückt man den Joystick nach oben, so springt Willi.

Vorschläge

Diese Version von Magic Forest verfügt über fünf Stufen. Der Autor wird wahrscheinlich später einmal eine verbesserte Version schreiben. Sie können auch mit AMOS TAME Ihre eigenen Stufen erzeugen und hinzufügen. Wir würden uns wirklich freuen, wenn Sie uns Ihre Änderungen dann auch zusenden, damit wir sie uns ansehen können!

Entwickelt von:

Programmierung und Spieldesign: Aaron Fothergill
Grafik und Spielidee: Adam Fothergill
Musik: Aaron und Adam Fothergill
Rettung in letzter Minute: Richard Vanner (Die A500-Besitzer haben dieses Spiel gerade noch bekommen! Spitzenleistung, Aaron!)

Number Leap

Frotzel, der Frosch hing ziemlich durch: er konnte das Einmaleins in der Schule überhaupt nicht in seinen Kopf kriegen. Und das machte ihm schwer zu schaffen: manchmal spielte er krank, damit er nicht zu gehen brauchte, bis dann seinem Vetter Ralf, dem rasenden Rechnungsprüfer, ein lustiger Weg einfiel, mit dem er ihm helfen konnte.

Ralf hob einen besonderen Teich aus und pflanzte Seerosen mit Zahlen darauf. Nun muß Frotzel nur noch ein Einmaleins aussuchen und die in diesem Einmaleins enthaltenen Zahlen erscheinen. Wenn er auf eine Seerose mit einer Zahl springt, die nicht in dieses Einmaleins gehört, so geht er unter. Schafft er's bis zur anderen Seite, so macht er vor Freude Riesenluftsprünge zurück!

Steuerung

Frotzel kann entweder mit dem an Port 1 (nicht dem Maus-Port) angeschlossenen

Joystick oder über die Cursor-Tasten gesteuert werden. Ein Einmaleins kann man wählen, indem man Frotzel auf die entsprechende Seerose stellt und den Feuerknopf (oder - bei Einsatz der Cursor-Tasten - die Leertaste) drückt. Wenn Sie während des Spiels eine beliebige Taste (außer den Cursor-Tasten) drücken, so sagt der Computer die Zahl der Seerose, auf der Frotzel sitzt, laut.

Entwickelt von:

Programmierung und Spieldesign: Peter J. Hickman
Grafik und Spielidee: C.D. White und Peter J. Hickman



28: AMOS Public-Domain-Library

Adresse:

c/o Sandra Sharkey
25 Park Road
Wigan
Lancashire
WN6 7AA
England

Telefon: 0044-942-495261

Und wie funktioniert's?

Die AMOS PD-Bibliothek ist eine besondere Disketten-Bibliothek, die Programme und Utilities umfaßt, die für AMOS-Anwender von Interesse sind.

Alle Programme enthalten unverschlüsselten Code, Sie können sie also genau untersuchen und lernen, wie man bestimmte Effekte erzielen kann.

Es ist auch ein Katalog verfügbar, in dem alle aktuellen Programme aufgeführt sind. Wenn Sie ein Exemplar möchten, so senden Sie bitte einen adressierten, entsprechend frankierten Rückumschlag an die obengenannte Adresse. Die Bibliothek ist jedem kostenlos zugänglich, ein Mitgliedsbeitrag wird nicht erhoben.

Alle Library-Disketten sind bootfähig. Enthalten sie lauffähige AMOS-Programme, so laufen diese Programme sofort ab. Wenn die Diskette mehrere Programme enthält, so erscheint ein Datei-Requester. Das AMOS-PD-Library-Logo mit der Adresse erscheint gleich nach dem Booten, so müssen Sie nicht ständig nach dem kleinen Zettel suchen, auf dem Sie sie notiert haben!

Wenn möglich arbeiten alle Disketten mit der neuesten Version des jeweiligen Programms, einschließlich Musikmodul und RAMOS. Damit auch VirusX zufrieden ist, wird der normaler Boot-Block eingesetzt.

Bitte vergessen Sie nicht, daß die AMOS PD-Library nicht Teil des AMOS-Clubs ist, und wir deshalb auch keine Fragen zum AMOS-Club beantworten können.

Zahlungsbedingungen

Auslandszahlungen müssen in Pfund Sterling entweder als internationale Geldanweisung, Euroscheck oder Postüberweisung geleistet werden. Wir können keine Fremdwährungen oder Kreditkarten annehmen. Die Zahlung kann auch in Form von britischen Briefmarken erfolgen, um die Gebühr für Postanweisungen zu umgehen.

Der Empfänger für Verrechnungsschecks/Postanweisungen ist die AMOS Public Domain Library (Adresse siehe oben).

APD-Disketten kosten £ 2,50 für Großbritannien, £ 2,75 für Europa und £ 3,— für alle anderen Länder. Porto und Verpackung sind im Preis enthalten.

Wenn Sie uns Ihre eigenen leeren Disketten schicken, so können Sie von diesem Preis £ 1,00 abziehen.

Wenn Sie mehr als zwei Disketten kaufen, so können Sie pro Diskette 20 Pence (einschließlich der ersten drei) abziehen, und auf diese Art mindestens 60 Pence sparen.

Leitfaden für das Einsenden von Disketten

Wenn Sie ein umwerfendes Spiel geschrieben haben, dann möchten Sie es uns vielleicht schicken, damit wir es in die Bibliothek aufnehmen können. Deshalb wollen wir Ihnen hier ein paar Tips geben, die Ihnen und uns die Sache erleichtern sollen.

Bitte bringen Sie auch die Anleitung - falls erforderlich - auf der Diskette, die Ihr Programm enthält, unter. Vergessen Sie nicht, extra anzugeben, wenn Ihr Programm mehr als 512 KByte benötigt.

Überprüfen Sie Ihre Disketten stets auf Viren, bevor Sie sie einsenden und vergessen Sie nicht, eine Diskette aus dem Katalog auszusuchen, die Sie im Tausch gegen Ihre haben möchten. Wenn Sie eine der lizenzierten Disketten möchten, so müssen Sie auch die Lizenzgebühr mitschicken, damit wir sie an den Autor weiterleiten können.

Lizenzware

Die Disketten in diesem Bereich kosten etwas mehr als die normalen PD-Disketten, da der Autor noch zusätzlich eine Lizenzgebühr erhält. Es handelt sich hier nicht um Public-Domain-Software, und das Kopieren und Verteilen dieser Disketten ist nicht erlaubt.

LPD1: *Colouring Book (Malbuch)* - von Trevor Prince. Einfaches Spiel für Kinder ab dem Vorschulalter. Man lädt eines der sechs zur Wahl stehenden Bilder, die beliebte Kinderreime darstellen, klickt die gewünschte Farbe an und malt das Bild aus. Zur Zeit wird an einer neuen Version gearbeitet, die einen Skizzenblock, Animation und noch vieles mehr umfaßt. Läuft auf Maschinen mit 1/2 MByte - Preis: £ 3,50 GB, £ 3,75 Europa und £ 4,— alle anderen Länder.

LPD4: *THINGAMAJIG von Len Tucker.* Spielen Ihre Kinder (oder Sie) gerne Puzzle? Dann ist dies das richtige Spiel für Sie. Die Diskette enthält 24 Bilder, unter denen Sie wählen können. Sie haben zwei Optionen, leicht oder schwer (EASY/HARD) - und das ist enorm schwer! Dann sehen Sie das Bild Ihrer Wahl als Strichzeichnung und müssen die Teile am richtigen Platz einfügen. Wenn alle Stricke reißen, gibt es noch eine Hilfe-Funktion. Ein ausgefeiltes, professionelles Programm. Läuft auf Maschinen mit 1/2 MByte - Preis: £ 3,50 GB, £ 3,75 Europa und £ 4,— alle anderen Länder.

LPD5: *JUNGLE BUNGLE von Len Tucker.* Nur für 1-MByte-Maschinen. Ein für Kinder gedachtes, icongesteuertes Abenteuerspiel, das jedoch auch die anspruchsvollsten Spieler in seinen Bann schlägt. Schöne Grafik mit Sample-Sound und etwas Animation. Es macht Spaß, Regentropfen aufzufangen, aber man muß sich vor dem Bananenklau in Acht nehmen! Preis: £ 3,50 GB, £ 3,75 Europa und £ 4,— alle anderen Länder.

LPD7: *4 WAY LYNX von Andreas Andreou* - 1 MByte und Joystick erforderlich. Ausführliche Dokumentation auf der Diskette. Das Ziel dieses 22 Stufen umfassenden

Puzzle-Spiels besteht darin, eine bestimmte Anzahl von Versatzstücken mit dem Mittelstück zu verbinden. Man muß jeweils erst die Zielvorgabe für eine Stufe erfüllen, bevor man zur nächsten Stufe fortschreiten kann. Sie können auch selbst Stufen erzeugen, die Höchstanzahl liegt bei 40. Die Stufen werden zunehmend schwieriger. Achtung, bei diesem Spiel wird man leicht süchtig! Preis: £ 3,50 GB, £ 3,75 Europa und £ 4,— alle anderen Länder.

Die verfügbaren Programme

Die AMOS PD-Sammlung wurde in 6 verschiedene Kategorien eingeteilt, um es Ihnen zu erleichtern, die Disketten zu finden, die Sie am meisten interessieren. Wir führen hier nur einige der Disketten des Katalogs auf, insgesamt sind nämlich mehr als 100 Disketten erhältlich!

Utilities

Einige dieser Utilities stammen aus der allgemeinen Public-Domain-Software, während andere in AMOS geschrieben wurden. Sie sind alle für AMOS- Programmierer von Nutzen.

APD1: *GAMES MUSIC CREATOR* - eine leistungsstarke und bedienerfreundliche Utility, mit der Sie Melodien erzeugen können. Greift auf alle vier Kanäle zu und umfaßt DOC-Dateien. Bearbeiten Sie einen "Block" von Musikdaten mit dem GMC- to-AMOS Konvertierprogramm und spielen Sie die Musik dann in AMOS einfach mit einem MUSIC-Befehl ab.

APD6: *STOS TO AMOS UTILITY* - Dieses Programm ermöglicht die Konvertierung von Atari ST- und IBM-Dateien. Umfaßt IFF-Konverter und andere AMOS STOS-Konvertierdateien.

APD7: *VIRUSX AND OTHER UTILITIES* - VIRUSX 4.1 und noch weitere Utilities wurden auf dieser Diskette untergebracht. Ein sinnvoller Bestandteil jeder AMOS- Sammlung.

APD31: *SCREEN DESIGNER von James Robert Crosby* - Dieses Programm soll es AMOS-Anwendern ermöglichen, große IFF-Schirme zu erzeugen, die dann in ihre eigenen Programme geladen werden können. Darüberhinaus enthält diese Diskette eine Vielzahl weiterer Routinen und Utilities, die in AMOS geladen werden können.

APD76: *RAINBOW WARRIOR von Martyn Brown* - Umfassende Dokumentation auf der Diskette. RAINBOW WARRIOR gibt Ihnen die Möglichkeit, mit Copper/Regenbogen zu "malen" und sie zur Integration in Ihre eigenen Programme zu speichern. Das Programm gibt ASCII-Blöcke aus, somit können sie direkt kombiniert werden. RAINBOW WARRIOR ist in AMOS geschrieben, so kann der Anwender auch seine eigenen Programme anpassen, mischen und kombinieren. Außerdem können Daten für K-Seka, Devpac, im Binär-, Dezimal, Hexadezimal- oder den eigenen Formaten ausgegeben werden. Die Diskette enthält die neuesten Versionen der besten AMOS-Utilities z.B. die neuesten Soundtrack-Konverter, Squash-a-bob und RAMOS. Sogar die Datenbank "BOMBASE" von Gareth Lancaster findet sich hier. Diese Diskette ist für alle Profi-Anwender ein

absolutes Muß.

APD83: *AMOS PAINT von Jounii Poullnen* - Läuft auf 1/2 MByte, 1 MByte ist jedoch empfehlenswert. Es gibt zwei Versionen von AMOS-Paint, PAL oder NTSC. Als Default ist die PAL-Version eingesetzt, aber durch Editieren der Startup- Sequenz der AMOS Paint-Diskette kann man das ändern. Verfügt über folgende Eigenschaften: 2-64 Farben (Sie können HAM-Bilder laden), bedienerfreundliche Schnittstelle mit automatisch verborgenem Menü, umfassende Malfunktionen, glattes Echtzeit-Autoscrollen auf Super-Bitmaps, sowohl IFF- wie auch mit SPACK gepackte Dateien können gespeichert und geladen werden, Farbwechsel mit SHIFT UP oder SHIFT DOWN, alle üblichen Mal-Tools, Ausmalen bestimmter Bereiche.

AMOS-Programme und -Spiele

Einer der einfachsten Wege, ein Programm zu lernen besteht darin, sich die Arbeit eines anderen anzusehen. Mit den AMOS PD-Disketten wird es kinderleicht!

APD2: *TREASURE SEARCH von Peter Hickman* - Die Schatzsuche steht im Mittelpunkt dieses ausgetüftelten Lernprogramms. Durch überragenden Sound und Grafik macht dieses Spiel allen Spaß, von 2 bis 92!

APD21: *WORD SQUARE SOLVER + GAMES* - Mit diesem Programm sind Kreuzworträtsel ein Kinderspiel. Man gibt nur die Buchstaben ein und der Computer bildet die richtigen Wörter. Außerdem enthält die Diskette noch die Spiele: Demolition Mission, Grub Grabber, Space Invaders, Pacman und ein U-Boot-Spiel.

APD32: *AMOS PROGRAMS 1 von Gary Fearn und Nadeem* - Enthält CLI.AMOS, Myfirstdemo.AMOS, Spritestars.AMOS und Textview.AMOS von Gary. Es ist auch von Nadeem eine Demo auf der Diskette. Sie finden hier eine Menge Quelltext, an dem Sie experimentieren und lernen können.

APD54: *AMOS PROGRAMS 2 von Gary Shilvock und Jason Anthony* - Diese Diskette umfaßt eine Zusammenstellung von Utilities, Routinen und Demos. Noch mehr Quelltext, mit dem Sie arbeiten können.

APD62: *ARCADIA* - Bei 1/2 MByte zweites Laufwerk entfernen. Das ist eines meiner Lieblingsspiele. Fesselnde Version des beliebten Arkanoid-/Breakout-Spiels. Als Zugabe bekommen Sie sogar einen Stufen-Editor, somit können Sie das Spiel nach Herzenslust verändern. Sehr empfehlenswert.

APD85: *REVERSI & AMOS SNAKES* - Reversi ist eine absolut begeisternde Version von Othello. Ich bin absolut süchtig danach! Man kann zu zweit oder gegen den Computer spielen. Außerdem ist eine ausgezeichnete Version des beliebten Spiels "Snakes & Ladders" (so ähnlich wie Mensch-ärgere-dich-nicht) auf der Diskette. Meine Kinder haben um den ersten Platz gekämpft, während ich noch nicht einmal eine sechs gewürfelt hatte!! Ich habe eben einfach kein Glück im Spiel. Zwei unterhaltsame Spiele auf einer Diskette. Sehr empfehlenswert. Beide laufen auf 1/2-MByte-Maschinen.

APD97: *DYNAMITE DICK von Adam Leech* - Ein ganz nettes kleines Spiel, in dem Sie den Schatz entdecken, die Monster in die Luft jagen und den Schlüssel zur nächsten Stufe finden müssen. Eine Diaschau und Demo vom selben Programmierer ist mit auf der Diskette. Empfehlenswert.

Demos

Demos eignen sich ausgezeichnet als Quelle für gute Routinen und Sie haben Glück, denn Sie können hier nach Belieben zugreifen.

APD9: *AMOS BIG DEMO V4 von Peter Hickman* - Diese Demo wurde von Peter Hickman geschrieben, um AMOS zu präsentieren. Sie wurde zwar ursprünglich mit einer älteren Version von AMOS erstellt, ist aber nach wie vor äußerst beeindruckend. Die Diskette enthält auch einen Ausschnitt aus dem Bericht über ein Interview, das Martyn Brown mit François Lionet führte - ganz interessant.

APD22: *FUN SCHOOL 3 Demo von Peter Hickman* - Die dritte Version der ausgezeichneten Lernreihe “Spielend Lernen”. Entspannen Sie sich und lassen Sie sich von Teddy durch die Spiele für die drei Altersgruppen führen. Diese Diskette zeigt, wie nützlich AMOS bei der Erstellung von Software-Demos sein kann.

APD81: *JUKEBOX DEMO* - Läuft nur zusammen mit APD82. Musik auf zwei Disketten. Diese Demo wurde von Mandarin Software für die AMOS-Werbung eingesetzt.

APD82: *JUKEBOX DEMO* - Läuft nur zusammen mit APD81. Diskette 2.

APD99: *BENSON DEMO 1 von Leslie Benzies* - Der Gewinner eines von Mandarin veranstalteten Wettbewerbs. Enthält ganz ausgezeichneten Quelltext für das Scrollen usw. Empfehlenswert.

APD100: *AMOS BIG DEMO II von Peter Hickman.* Mit dieser Demo hat Mandarin AMOS auf einigen Computer-Messen präsentiert. Enthält ganz ausgezeichneten Quelltext, aus dem Sie viel lernen können. Äußerst empfehlenswert.

Grafik- und Fonts-Diskette

PD78: *IFF Picture Disc #4* - Eine Auswahl an IFF-Bildern aus den Pressearchiven. Auf dieser Diskette sind einige atemberaubende Bilder, zum Beispiel ein süßes Kätzchen auf einem Boot, eine Giraffe, Jack Nicholson als der Joker, und ein toller Frosch - insgesamt zwanzig Bilder und Musik. Ideal, wenn Sie nicht zeichnen können und mit IFF-Bildern in AMOS experimentieren möchten.

APD3: *FONTS DISC #1* - Enthält 14 Schriftarten, einschließlich TIMES, BOOKMAN, HELVETICA und TINY.

APD4: *FONTS DISC #2* - 13 Schriftarten, einschließlich AVANT GARDE, CELTIC,

PALO ALTO, BASEL, PEIGNOT und ALDUOUS.

APD5: *FONTS DISC #3* - 14 Schriftarten, einschließlich BROADWAY, CAMELOT, FUTURE, STENCIL und VANCOUVER.

APD38: *IFF FONTS DISC* - Eine Auswahl von IFF-Fonts, die von "The Skunk" aus dem ST-Format konvertiert wurden, und einige von "Spadge" hinzugefügte, ausgefeilte AMIGA-Fonts. Sie können in BOBs zerlegt und mit AMOS eingesetzt werden. Alle Fonts werden in einer Diaschau mit Musik präsentiert.

AMOS-Musik

Alle Diskette booten, und die Musik kann in AMOS abgespielt werden. Unter "Music General" - Musik allgemein - und "Australian Collection" finden Sie weitere Samples-Disketten, ST-Module und Instrumente.

Wir können natürlich hier im Handbuch nicht alle verfügbaren Disketten aufführen, aber Sie finden sie vollzählig im AMOS PD-Library-Katalog.

Musik allgemein

Wir verfügen über eine Reihe von Soundtracker-Modulen, Instrumenten und Samples, die Sie mit GMC etc. einsetzen können. Nähere Einzelheiten dazu können Sie dem Katalog entnehmen.

Alle Samples wurden als IFF-Samples belassen, so daß sie über die Utility zur Sample-Erzeugung in AMOS eingesetzt werden können. Alle Samples-Disketten booten, haben ein Inhaltsverzeichnis und der Anwender kann sie über eine kleine Utility anhören.

Die Instrumentdisketten sind für den Einsatz mit GMC und der SOUND/NOISE/PROTRACKER-Serie bestimmt, und verwenden deshalb das konventionelle "ST"-Bezeichnungssystem. Jede der Disketten weist eine vorprogrammierte Liste auf, und es besteht die Möglichkeit, alle Samples über Perfect Sound anzuhören.

Alle Moduldisketten booten und zeigen Information an. Der Anwender kann die Module mit dem PD-Intuitracker-Paket anhören. Außerdem wurden sie zu ABK- Dateien konvertiert und erscheinen auf den Music.ABK-Disketten.

29: AMOS-Fehlermeldungen

Editor-Fehlermeldungen

Die folgenden Meldungen können manchmal in der Statuszeile erscheinen, wenn Sie Ihre Programme editieren.

Anfang des Textes: Der Text-Cursor ist oben an Ihrem Programm angekommen. Das Drücken der Pfeiltaste nach oben hat nun keine Wirkung mehr.
Ende des Textes: Der Cursor hat die letzte Zeile Ihres aktuellen Programms erreicht.
Kein Pufferspeicherplatz mehr: Sie haben keinen Platz mehr im Editor-Bereich. Speichern Sie Ihr Programm auf Diskette und erweitern Sie den Puffer mit dem Befehl TEXTPUFFER (im SUCH-Menü). Wenn Sie denn immer noch Probleme haben, ist es möglicherweise nötig, daß Sie Ihr Programm in mehrere Teile aufspalten und sie nacheinander mit dem RUN-Befehl aus AMOS-Basic ablaufen lassen.
Kein Speicherplatz mehr: Es ist für Ihre Programme nicht mehr genug Speicher vorhanden. Versuchen Sie, über CLOSE WORKBENCH die 40 KByte freizusetzen, die der Amiga Workbench-Schirm verbraucht.
Keine Fehler: Während des Tests wurden keine Fehler in Ihrem Programm entdeckt.
Keine Procedure: FALTEN funktioniert nur, wenn der Cursor auf einer Prozedur steht.
Nicht gefunden: Der zuvor aufgerufene Suchbefehl war ohne Erfolg.
Nicht markiert: Sie können die Markierung nicht verschieben, weil keine Markierungen gesetzt sind.
Programm paßt nicht in Textpuffer: Diese Meldung erscheint, wenn der Platz, der für das Laden eines Programms erforderlich ist, den im Puffer des Editors verfügbaren Bereich übersteigt. Wenn Sie die nachfolgende Aufforderung von AMOS mit *Ja* beantworten, so wird der Textpuffer genau auf die Größe gestellt, die das zu ladende Programm aufweist. Das Programm können Sie nicht erweitern, sondern nur seine Größe reduzieren. Falls erforderlich, können Sie im SUCH- Menü *Textpuffer* aufrufen, um den Textpuffer zu erweitern. Wenn Sie *Nein* wählen, wird der Lade-Vorgang abgebrochen und der mindestens erforderliche Puffer in der Statuszeile angezeigt.
Syntax Fehler: Die Syntax (Grammatik) der aktuellen Zeile weist einen Fehler auf. Das korrekte Format können Sie dem Handbuch oder der Referenzkarte entnehmen.
Variablenpuffer zu klein: AMOS verfügt über einen Puffer, der die Namen all Ihrer Variablen enthält. Wenn Sie zuviele lange Namen generieren, wird diese Fehlermeldung angezeigt. Sie können die Defaultgröße des Puffers über das CONFIG-Zusatzprogramm verändern.
Welcher Block?: Sie müssen einen Block erst definieren, bevor Sie ihn ausschneiden oder einsetzen (CUT/PASTE) können.
Zeile zu lang: Der AMOS-Editor kann nur Zeilen bearbeiten, die maximal 255 Zeichen lang sind.
Zu viele Variablen im Direkt Modus: In dem Direkt-Modus ist nicht genug Platz für Ihre Direkt-Modus-Variablen. Der Direkt-Modus erlaubt Ihnen das Erzeugen von bis zu 64 neuen Variablen, aber das kann eingeschränkt werden, wenn Ihr Programm zuviel Speicherplatz einnimmt.

Programmfehler

Wenn Sie eines Ihrer Programme ablaufen lassen oder es durch den TEST-Befehl auf dem MENU-Fenster überprüfen, so führt AMOS eine vollständige Überprüfung aller einzelnen Anweisungen durch. Auf diese Art können Sie die meisten Fehler direkt aus dem Editor heraus beseitigen, ohne erst Ihre Programme ablaufen lassen zu müssen. Hier ist eine vollständige Liste dieser Fehlermeldungen:

Befehl muß in einer Procedure verwendet werden: Der Befehl SHARED kann nur IN einer Prozedurdefinition eingesetzt werden.
DATA muß am Anfag der Zeile stehen: Alle DATA-Anweisungen in Ihrem Programm müssen ganz an den Anfang einer Zeile gestellt werden (außer Sprungmarkendefinitionen).
DO ohne LOOP: Die Anweisungen DO und LOOP müssen stets ein Paar bilden. Jede DO-Struktur muß durch einen LOOP-Befehl abgeschlossen werden.
ELSE ohne ENDIF: Sie haben den letzten ENDIF-Befehl in einer strukturierten IF-Abfrage vergessen.
ELSE ohne IF: Die ELSE-Anweisung kann nur in einer strukturierten Abfrage eingesetzt werden.
ENDIF ohne IF: Es wurde ein ENDIF-Befehl in Ihrem Programm gefunden, der keiner passenden IF-Anweisung entspricht.
Falsche Anzahl an Parametern: Sie haben versucht, eine falsche Anzahl von Werten in eine Anweisung oder eine Prozedur einzugeben.
Falsche Struktur: Alle geschachtelten Schleifen müssen VOLLSTÄNDIG ineinander gestellt werden. Es ist illegal, daß sich Schleifen "kreuzen". Beispiel:

```
Do
   If A=B
Loop
   Print A
Endif
```

Das ist nicht erlaubt.

Feld ist im Hauptprogramm nicht definiert: Sie haben versucht, auf ein Array in einer Prozedur zuzugreifen, das im Hauptprogramm nicht dimensioniert wurde.
Feld noch nicht definiert: Das Element, das Sie in dem Ausdruck angegeben haben, gehört keinem der zuvor dimensionierten Arrays an.
Feld wurde schon definiert: Sie können das gleiche Array nicht zweimal in Ihrem Basic-Programm dimensionieren.
FOR ohne passendes NEXT: Ein FOR-Befehl steht ohne die zugehörige NEXT-Anweisung im Programm.
IF ohne ENDIF: Die Anweisungen in einer strukturierten IF-Abfrage sollten stets durch einen ENDIF-Befehl abgeschlossen werden. Verwechseln Sie diese Abfragen bitte nicht mit dem Befehl IF..THEN - da besteht ein großer Unterschied!
Kein Speicherplatz mehr: Dieser Fehler kann während des Tests auftreten, wenn AMOS den erforderlichen Variablen-Puffer aufgrund von Speicherknappheit nicht reservieren kann (siehe SET BUFFER).

Keine Sprünge in der Mitte einer Schleife!: Sie können mit einer GOTO- oder GOSUB-Anweisung nicht direkt in eine Schleife hineinspringen, wenn Sie sich jedoch in einer Schleife befinden, können Sie so herausspringen.
Leere Klammern zu SHARED-Definition eines Feldes nutzen: Wenn Sie ein Array als SHARED definieren möchten, können Sie dazu zum Beispiel folgenden Befehl einsetzen:

```
Shared Array()
```

Es ist nicht zulässig, hier auch die Dimensionen des Arrays hinzuzufügen.
LOOP ohne DO: Es wurde ein LOOP-Befehl entdeckt, der keiner entsprechenden DO-Anweisung zugeordnet werden kann.
NEXT ohne FOR: AMOS ist auf eine NEXT-Anweisung gestoßen, der kein vorausgehender FOR-Befehl zugeordnet werden kann.
Nicht genug Schleifen zum verlassen: Der Schleifenzähler, den Sie in einem EXIT- oder EXIT IF Befehl angegeben haben, übersteigt die Anzahl der aktiven Schleifen.
Procedure nicht definiert: Die Prozedur, die Sie aufgerufen haben, existiert in Ihrem Programm bisher noch nicht.
Procedure nicht geöffnet: Eine END PROC Anweisung wurde ohne zugehörige PROCEDURE-Definition angetroffen.
Procedure nicht geschloßen: Bei einer Ihrer Prozeduren fehlt die Anweisung END PROC.
Procedure-Deklaration muß alleine in der Zeile stehen: Die Anweisungen PROCEDURE und END PROC müssen in einer eigenen Zeile beginnen.
REPEAT ohne passendes UNTIL: Es gibt in Ihrem Programm eine REPEAT-Anweisung ohne zugehörigen UNTIL-Befehl.
SHARED muß alleine in der Zeile stehen: Der Befehl SHARED muß als einzige Anweisung in der aktuellen Zeile stehen.
Sprungmarke doppelt definiert: Jede Sprungmarke oder Prozedur kann in Ihrem Programm nur einmal definiert werden.
Sprungmarke nicht definiert: AMOS kann die Sprungmarke, die Sie in der Anweisung angegeben haben, nicht finden.
Syntax Fehler: Die Syntax (Grammatik) der aktuellen Zeile weist einen Fehler auf. Das korrekte Format können Sie dem Handbuch oder der Referenzkarte entnehmen.
THEN darf nicht in einer strukturierten Abfrage stehen: Der Befehl IF...THEN kann in einem strukturierten Text nicht eingesetzt werden. Verwenden Sie stattdessen IF...ENDIF.
UNTIL ohne REPEAT: Dem UNTIL-Befehl entspricht keine vorhergegangene REPEAT-Anweisung in Ihrem Programm.
Variable ist schon als SHARED definiert: Sie können dieselbe Variable nur einmal in einer Prozedur definieren.
Variablen Puffer muß am Anfang gesetzt werden!: Der Befehl SET BUFFER muß stets in der ersten Zeile Ihres Programms stehen (einschließlich der Bemerkungen - REMs).
Variablen Puffer zu klein: Während ein Programm getestet wird, verwendet AMOS einen Speicherbereich, der für die Variablen reserviert ist. Sie erhalten diese Fehlermeldung, wenn der Pufferbereich überschritten wird. Wenn Sie genug Speicher

zur Verfügung haben, können Sie durch den Befehl SET BUFFER mehr Speicher für Ihre Variablen abrufen.
WEND ohne WHILE: Dem WEND-Befehl entspricht keine vorausgegangene WHILE-Anweisung in Ihrem Programm.
WHILE ohne passendes WEND: AMOS kann keine WEND-Anweisung finden, die zu dem aktuellen WHILE gehört.
Zusatz nicht geladen: Sie haben versucht, ein Programm ablaufen zu lassen, das einen der neuen Befehle aus einer Erweiterungsdatei enthält. Die entsprechenden Erweiterungen müssen erst auf Ihrer Bootdiskette installiert werden und zum Einsatz über das CONFIG-Zusatzprogramm ausgewählt werden.

Run-Time-Fehler

Stößt Ihr AMOS-Programm beim Ablaufen auf einen Fehler, so werden die folgenden Fehlermeldungen generiert. Dann bricht AMOS Ihr Programm ab und hebt die aktuelle Anweisung hervor. Wenn Sie in den Editor zurückkehren, so wird der Cursor sofort in die betreffende Zeile gestellt.

Wenn Sie Korrektur-Routinen einsetzen, so möchten Sie vielleicht die Fehlermeldung erhalten, die einer bestimmten Nummer in Ihrem System entspricht. Dazu setzen Sie zum Beispiel folgende Zeile ein:

```
Error Errn
```

Jeder Run-Time-Fehler verfügt über eine eigene Fehlernummer, die unten aufgeführt ist.

Durch einen besonderen FOLLOW-Befehl können Sie auch sehen, wie sich Ihre Variablen im Laufe Ihres Programmes verändern. Hier sind nun die FOLLOW-Befehle aufgeführt:

FOLLOW *(Einsicht in den Status einer oder mehrerer Variablen)*

FOLLOW [Reihe von Ausdrücken...]

Der Follow-Befehl hält den Programmablauf an, bis der Anwender auf die Leertaste drückt. Werden bei der Eingabe des Befehls Ausdrücke spezifiziert, so werden ihre Ergebnisse in einem gesonderten Fenster angezeigt. Ähnlich wie das Direkt-Modus-Fenster hat auch dieses Fenster keine Auswirkungen auf Ihren Programmschirm. Sie können das Fenster auf der Anzeige einfach mit den Cursor-Tasten auf- und abbewegen.

Gefaltete Prozeduren werden von diesem System nicht erfaßt, so daß Sie nur die Prozedur oder Routine bearbeiten können, die untersucht werden soll.

FOLLOW OFF *(Hebt vorausgegangenen Follow-Befehl auf)*

FOLLOW OFF
Durch diesen Befehl wird jede vorausgegangene Follow-Anweisung gelöscht und das FOLLOW-Fenster aus der Anzeige entfernt.

Addressen Fehler (25): Diese Meldung erscheint, wenn eine Adresse in einem DOKE-, DEEK-, LOKE- oder LEEK-Befehl eingesetzt wird, die einen ungeraden Wert darstellt.
Animationskette zu lang (113): Das aktuelle AMAL-Programm übersteigt die maximale Größe von 65536 Bytes. Versuchen Sie, das Programm in kleinere Einheiten zu spalten. Es ist völlig zulässig, das gleiche Objekt über verschiedene AMAL-Kanäle zu animieren.
Autotest schon aktiviert (111): Ein AMAL-Autotest wurde in einem anderen Autotest-Befehl definiert. Das ist natürlich nicht erlaubt.
Bank noch nicht reserviert (36): Diese Fehlermeldung zeigt an, daß die Bank, die Sie gewählt haben, nicht mit dem Befehl RESERVE erzeugt wurde. Die Meldung kann auch nach Befehlen wie PASTE ICON oder SAMPLAY auftreten, durch die automatisch Informationen aus einer bestimmten Speicherbank geladen werden.
Bank schon reserviert (35): Sie haben versucht, eine Speicherbank zu erzeugen, die bereits existiert. Beachten Sie, daß die Banken 1 bis 4 normalerweise jeweils für Ihre Sprites, Icons, Musik und Menüdefinitionen bestimmt sind.
Befehl läuft nur im Autotest (112): Die Befehle Direct und eXit können nur in einem AMAL-AUTOTEST eingesetzt werden.
Bild paßt nicht in aktuellen Schirm (32): Sie haben versucht, mit LOAD IFF ein Bild in einen vorhandenen Schirm zu laden, aber der Typ der beiden Schirme stimmt nicht überein. Hängen Sie die Nummer des Zielschirms folgendermaßen an den Befehl LOAD IFF an:

```
Load Iff "Dateiname",Nummer
```

Vorausgesetzt, der Schirm, den Sie durch den Parameter Nummer angeben, liegt im richtigen Bereich (0-7), so wird AMOS während des Ladevorgangs automatisch einen Schirm der erforderlichen Art erzeugen.
Block nicht gefunden (65): Der Block, den Sie in dieser Anweisung angegeben haben, ist nicht durch GET BLOCK erzeugt worden.
Bob nicht definiert (68): Das BOB, das Sie manipulieren wollten, wurde nicht durch einen vorausgegangenen BOB-Befehl erstellt. Diese Meldung kann nach einem Fehler in der Anweisung PASTE BOB auftreten.
Copper Liste zu lang (77): Als Default ist Ihre anwender-definierte Copperliste auf maximal 12 KByte beschränkt. Sie kann jedoch über eine Option aus dem CONFIG-Zusatzprogramm erweitert werden.
Copper nicht desaktiviert (76): Sie haben versucht, die Befehle COP MOVE oder COP SWAP einzusetzen, ohne zuvor die normale Copperliste durch COPPER OFF außer Kraft zu setzen.
Datei existiert schon (79): Es ist nicht möglich, eine Datei durch RENAME auf einen Dateinamen oder ein Verzeichnis umzubenennen, die bereits existieren.
Datei gegen Lesen geschützt (91): Die gewünschte Datei hat einen Leseschutz. Nähere Einzelheiten zu dem Befehl PROTECT, der im CLI verfügbar ist, finden Sie im Amiga Handbuch, das Sie zusammen mit Ihrem Computer erhielten.
Datei gegen Löschen geschützt (89): Durch das Amiga Diskettensystem können Sie einzelne Dateien mit dem speziellen PROTECT-Befehl aus dem CLI vor dem Löschen schützen. Wahrscheinlich haben Sie versucht, eine wichtige Datei mit DELETE zu löschen.

Datei gegen Überschreiben geschützt (90): Sie können die gewählte Datei nicht verändern, da Sie durch den PROTECT-Befehl (CLI) geschützt ist.
Datei nicht gefunden (81): Sie haben versucht, auf eine Datei oder ein Verzeichnis zuzugreifen, die im aktuellen Verzeichnis nicht vorhanden sind.
Datei nicht geöffnet (97): Ihr Programm hat versucht, Daten von oder an eine Datei zu übertragen, die nicht zuvor mit OPEN IN, OPEN OUT, APPEND usw. geöffnet wurde.
Dateiformat nicht erkannt (95): Durch den LOAD-Befehl erhält man nur Zugang zu den AMOS-Speicherbanken auf der Diskette. Sie haben vielleicht LOAD mit LOAD IFF verwechselt (lädt einen Schirm). Wenn Sie eine im normalen Amiga-Format gespeicherte Datendatei laden möchten, so sollten Sie stattdessen die BLOAD- Anweisung einsetzen.
Device nicht ansprechbar (86): Die in der Anweisung angegebene Diskette oder das Device sind nicht mit dem Amiga verbunden. Diese Fehlermeldung wird oft durch einen unerwarteten Diskettenwechsel hervorgerufen. Die Lösung besteht darin, das Directory zum Beispiel mit der folgenden Zeile auf das entsprechende Laufwerk zu stellen:

```
Dir$="Df0:"
```

Sollte der Fehler auftreten, wenn Sie mit dem Datei-Requester arbeiten, so klicken Sie eine der Laufwerkstasten wie "Df0:" an und wechseln Sie über die Taste Set Dir das Laufwerk.
Disk nicht datentragend (83): Wenn Sie eine Diskette einlegen, so überprüft der Amiga stets, ob sie zulässig ist. Tritt hier ein Fehler auf, so erhalten Sie diese Meldung. Aufgrund der Komplexität des Diskettensystems des Amiga werden diese Fehler manchmal versehentlich hervorgerufen. Stellt sich das Problem als hartnäckig heraus, so müssen Sie vielleicht das DISC DOCTOR Programm auf der normalen Workbench-Diskette einsetzen. Diese Fehlermeldung kann auch auftreten (das ist uns passiert), wenn Sie aus Versehen den Stecker ziehen und dadurch die Festplatte außer Betrieb setzen!
Disk schreibgeschützt (84): AMOS kann keine Information auf die Diskette speichern, weil diese mit einem Schreibschutz versehen ist. Schieben Sie also den Schreibschutz zur Seite oder verwenden Sie eine andere Diskette.
Diskette voll (88): Auf der aktuellen Diskette ist nicht mehr genug Platz für Ihre Daten.
Eingabe zu lang (99): Eine eingegebene Zeichenkette ist zu lang für eine zuvor definierte Variable. Oder Sie haben vielleicht versucht, über INPUT# eine Zeile einzugeben, die länger als 1000 Zeichen ist.
Ende der Datei (100): Während eines Zugriffs auf die Diskette wurde unvermittelt das Ende der aktuellen Datei erreicht. Mit der EOF-Funktion können Sie Ihr Programm nach diesem Zustand abfragen.
Ende des Programms (10): Diese Meldung wird angezeigt, nachdem AMOS die letzte Anweisung in Ihrem Programm ausgeführt hat.
Falsche Anzahl an Farben (49): Die Hardware des Amiga unterstützt nur bestimmte Farbkombinationen, die dann gleichzeitig auf dem Bildschirm erscheinen können. Eine vollständige Liste der verfügbaren Optionen finden Sie im Kapitel über Schirme (SCREEN-Befehle). Wahrscheinlich haben Sie sich beim Eingeben des Befehls SCREEN OPEN vertippt.
Falsche Block Parameter (66): In einem GET BLOCK oder einem PUT BLOCK Befehl ist ein Fehler. Die Werte, die Sie eingegeben haben, sind nicht zulässig.

Falsche Copper Parameter (78): Der Wert, den Sie in eine COP MOVE, COP MOVEL oder COP SWAP Anweisung eingesetzt haben, liegt außerhalb des erlaubten Bereichs.
Falsche Screen Parameter (48): Die Dimensionen, die Sie durch SCREEN OPEN spezifiziert haben, sind nicht zulässig. Die Mindestgröße des Schirms beträgt nur 32x8 und die Höchstgröße hängt ganz vom verfügbaren Chip-Speicher ab.
Falsche Window Parameter (60): Sie haben in einem Ihrer verschiedenen Window-Befehle einen falschen Wert eingegeben.
Falscher Befehl innerhalb Autotest (115): Sie haben versucht, einen AMAL-Befehl wie Move oder Anim in einem AUTOTEST einzusetzen. Überprüfen Sie die Schreibweise Ihrer AMAL-Sprungmarken. Vielleicht haben Sie den AUTOTEST versehentlich definiert.
Falscher Dateientyp (98): Ein Diskettenbefehl wurde eingesetzt, der in der aktuellen Datei nicht zulässig ist. Diese Meldung erhalten Sie zum Beispiel, wenn Sie versuchen, mit den GET- oder PUT-Befehlen auf eine sequentielle Datei zuzugreifen.
Falscher Dateiname (82): Sie haben versucht, einen Dateinamen zu verwenden, der den Bezeichnungsnormen nicht entspricht. Schlagen Sie dazu bitte im Bedienerleitfaden Ihres Amiga nach.
Falscher Funktionsaufruf (23): Diese Meldung wird hervorgerufen, wenn Sie bei der Eingabe der Werte in einen AMOS-Befehl einen Fehler machen. Eine vollständige Liste der zulässigen Parameter finden Sie im entsprechenden Abschnitt des Handbuchs.
Falsches IFF Format (30): Sie haben versucht, mit dem Befehl LOAD IFF eine Datei zu laden, die in einem ungewöhnlichen Format gespeichert wurde. Vergessen Sie nicht, daß LOAD IFF nur Schirme in den Speicher laden kann und nicht die allgemeinen IFF-Dateien.
Fehler nicht aufgearbeitet (3): Sie sind aus einer Routine zur Fehlerkorrektur ausgestiegen, ohne den Fehler durch RESUME zurückzusetzen.
Fehlerprocedure muß mit RESUME enden (8): Sie können aus einer Prozedur zur Fehlerkorrektur nicht mit END PROC aussteigen. Sie müssen stattdessen einen der speziellen RESUME-Befehle einsetzen.
Feld nicht definiert (27): Ihr Programm hat versucht, auf ein Array zuzugreifen, das nicht zuvor definiert wurde.
Feld schon definiert (28): Sie haben versucht, ein Array in Ihrem Programm zweimal zu dimensionieren. Normalerweise wird das im Laufe der Eingangsprüfung der Syntax erfaßt, aber wenn Ihr Programm kompliziert ist, so wird der Fehler nur entdeckt, wenn Sie tatsächlich versuchen, das Array neu zu dimensionieren.
Fenster hat keinen Rahmen (63): Sie haben versucht, den BORDER-Befehl für ein Fenster einzusetzen, das keinen Rahmen hat.
Fenster nicht geöffnet (54): Sie haben versucht, in ein Fenster zu gehen, das nicht existiert.
Fenster schon geöffnet (55): Sie haben versucht, ein Fenster zu öffnen, das bereits offen ist.
Fenster zu groß (57): Das gewünschte Fenster kann nicht geöffnet werden, weil es zu groß für den aktuellen Schirm ist.
Fenster zu klein (56): Des gewünschte Fenster ist zu klein. Die zulässige Mindestgröße beträgt 3x3.
FLASH - Deklarationsfehler (52): Sie haben in der Animationskette, durch die eine Farbsequenz mit FLASH definiert werden soll, einen Fehler gemacht.

Fonts nicht untersucht (37): Bevor Sie den Befehl SET FONT einsetzen können, müssen Sie mit den Befehlen GET FONTS, GET DISC FONTS oder GET ROM FONTS erst eine Liste der verfügbaren Fonts erzeugen.

Gültige Screen Nummern von 0-7 (50): AMOS läßt Sie nur höchstens acht Schirme gleichzeitig öffnen.

I/O error (94): Eine Ihrer Dateien scheint korrumpiert zu sein, und kann daher nicht richtig aufgerufen werden. Erweist sich das Problem als hartnäckig, so sollten Sie Ihre Laufwerksanschlüsse sorgfältig überprüfen. Eventuell müssen Sie Ihre Diskette mit dem DISK DOCTOR Programm auf der Original-Workbench-Diskette reparieren.

Icon nicht definiert (74): Das Icon, das Sie in Ihrer Anweisung angegeben haben, kann in der aktuellen Icon-Bank (Bank 2) nicht gefunden werden.

IFF Kompression nicht erkannt (31): Der Schirm, den Sie von der Diskette laden möchten, verwendet ein ungewöhnliches Kompressionssystem. Wenn möglich sollten Sie in das Grafikpaket zurückgehen, mit dem Sie den Schirm erzeugt haben, und ihn in einem normalen IFF-Format speichern.

Kann bei einer Sprungmarke nicht weitermachen (4): In einer Fehlerprozedur können Sie RESUME Label nicht einsetzen.

Kann Dual Playfield nicht aktivieren (70): Sie haben versucht, mit den falschen Schirmen ein Dual Playfield zu erzeugen. Im Abschnitt zum DUAL PLAYFIELD Befehl sind die zulässigen Kombinationen aufgeführt.

Kein Speicherplatz mehr (24): Das ist die normale Fehlermeldung, die erscheint, wenn Sie versuchen, den verfügbaren Speicher zu überschreiten. Nur keine Panik, denn es gibt die folgenden drei Möglichkeiten, Speicher zu sparen:

1 Durch CLOSE WORKBENCH die Amiga Workbench schließen, dadurch werden 40 KByte frei.
2 Durch CLOSE EDITOR das Editierfenster abschalten, wenn es nicht verwendet wird. Spart 24 KByte!
3 Wenn Sie der Statuszeile entnehmen können, daß noch genug freier Speicher vorhanden ist, so speichern Sie Ihr Programm und booten Sie den Computer nochmals. Dadurch wird das durch das Betriebssystem des Amiga hervorgerufene Problem der Speicherfragmentierung beseitigt.

Kein Stapelspeicher mehr (0): Diese Meldung wird generiert, wenn Sie versuchen, zuviele Prozeduren in einer Schleife aufrufen zu lassen. AMOS-Prozeduren können sich zwar selbst aufrufen (Rekursion), nach etwa 50 Schleifen erhalten Sie jedoch eine Fehlermeldung.

Kein Variablenspeicher mehr (11): Als Default weist AMOS Ihren Zeichenketten und Arrays nur 8 KByte Speicherplatz zu. Setzten Sie am Anfang Ihres Programms den Befehl SET BUFFER ein, um diesen Speicher nach Bedarf zu erweitern.

Keine AmigaDOS Disk (92): Außer Sie arbeiten mit einem Programm wie CROSSDOS, kann AMOS nur Disketten lesen, die auf dem Amiga erstellt wurden. Also rufen zum Beispiel PC- oder ST-Disketten diese Meldung hervor.

Keine Daten hinter der Sprungmarke (41): Durch RESTORE wurde versucht, den Datenzeiger auf eine Zeile zu setzen, die keine DATA-Anweisungen enthält.

Keine Disk im Laufwerk (93): Sie haben versucht, auf ein Laufwerk zuzugreifen, das keine Diskette enthält. Wenn Sie die Diskette gerade ins Laufwerk gelegt haben, so

warten Sie bitte ein paar Sekunden und versuchen es dann noch einmal.
Keine ON ERROR PROC vor diesem Befehl (5): RESUME LABEL ist nur nach einem ON ERROR PROC Befehl zulässig.
Keine weiteren Daten (33): Der READ-Befehl hat über die letzte DATA-Meldung in Ihrem Programm hinausgelesen. Wahrscheinlich haben Sie bei der Eingabe einer Ihrer DATA-Zeilen die Informationen nicht vollständig eingegeben. Überprüfen Sie auch die RESTORE-Befehle auf Tippfehler.
Keine ZONEn definiert (73): Bevor Sie SET ZONE einsetzen, müssen Sie der Zone zuerst durch RESERVE ZONE Speicherplatz zuweisen.
Menu nicht geöffnet (38): Der MENU ON Befehl wurde aufgerufen, aber das Menü fehlt. Sie müssen Ihr Menü erst mit der MENU$-Anweisung oder MAKE MENU BANK definieren.
Menu Punkt nicht definiert (39): Der Menüpunkt, den Sie in Ihrem Menübefehl spezifiziert haben, wurde nicht zuvor mit MENU$ definiert.
NEXT ohne FOR in Animationskette (108): Dies weist auf einen Fehler in einer Ihrer AMAL-Animationsketten hin. Jedem Next-Befehl muß eine For-Anweisung zugeordnet sein. Überprüfen Sie die Schreibweise der Anmerkungen in Ihren AMAL- Programmen.
Nicht als Zusatzprogramm definiert (43): Wenn das laufende Programm nicht als Zusatzprogramm installiert ist, und Sie versuchen, mit BGRAB auf eine Bank zuzugreifen, so erhalten Sie diese Fehlermeldung.
POP ohne GOSUB (2): POP kann nur in einer Subroutine ausgeführt werden, die zuvor durch GOSUB eingegeben wurde. Setzen Sie POP PROC ein, um aus einer Prozedur auszusteigen.
Programm nicht gefunden (42): Das im PRUN-Befehl genannte Programm wurde nicht zuvor in den Speicher des Amiga geladen. Arbeiten Sie mit Lade Andere aus dem MENU-Fenster.
Programm unterbrochen (9): Dies ist keine Fehlermeldung, Sie haben wohl entweder Ctrl+C gedrückt, oder sind mit einer STOP-Anweisung direkt aus Ihrem Programm ausgestiegen.
Rainbow nicht definiert (75): Bevor Sie in einem Ihrer Programme den RAINBOW-Befehl aufrufen, müssen Sie Ihren Regenbogeneffekt erst über SET RAINBOW definieren.
RESUME Marke nicht definiert (6): Die Sprungmarke, die Sie im RESUME-Befehl angegeben haben, existiert nicht.
RESUME ohne Fehler (7): Der RESUME-Befehl kann nur ausgeführt werden, wenn ein Fehler in Ihrem Programm aufgetreten ist. Am besten geht man nach einem Fehler in AMOS-Basic zurück.
RETURN ohne GOSUB (1): RETURN kann nur eingesetzt werden, um aus einer Subroutine auszusteigen, in die Sie ursprünglich durch GOSUB eingestiegen sind.
Screen ist schon im Double buffering (69): Sie haben versucht, DOUBLE BUFFER zweimal im selben Schirm aufzurufen.
Screen kann nicht animiert werden (67): AMAL kann Schirme nur bewegen oder scrollen. Man kann sie mit dem eingebauten Anim-Befehl nicht animieren.
Screen nicht geöffnet (47): Der Schirm, auf den Sie zugreifen wollen, wurde nicht zuvor mit dem SCREEN-Befehl geöffnet.
Screen nicht im Dual Playfield Modus (71): DUAL PRIORITY kann nur eingesetzt werden, nachdem Sie ein Dual Playfield erzeugt haben.

Scroll-Bereich nicht definiert (72): Bevor Sie den SCROLL-Befehl einsetzen, müssen Sie Richtung und Größe Ihres Scrollbereichs durch SET SCROLL definieren.
SHIFT - Deklarationsfehler (53): Sie haben in der in den Anweisungen SHIFT UP oder SHIFT DOWN eingesetzten Farbsequenz einen Fehler gemacht.
Sprite Fehler (105): Die Werte, die Sie in einen SPRITE-Befehl eingegeben haben, liegen nicht in den vorgegebenen Bereichen.
Sprung in/aus Autotest in Animationskette(109): Es ist nicht legal, aus Ihrem AMAL-Hauptprogramm heraus in einen AUTOTEST zu springen. AUTOTEST müssen Sie immer erst durch die Befehle Direct oder eXit verlassen.
Sprungmarke in Animationskette nicht definiert (114): Diese Meldung wird generiert, wenn Sie versuchen, in einer AMAL-Animationskette zu einer nicht vorhandenen Sprungmarke zu springen.
Sprungmarke nicht definiert (40): Die Sprungmarke in Ihrer Anweisung wurde in Ihrem Programm nicht definiert. Überprüfen Sie die GOTOs, GOSUBs oder RESTORE-Anweisungen auf Fehler.
Sprungmarke schon im Animationskette definiert (109): AMOS hat in Ihrem AMAL-Programm zwei Versionen der gleichen Sprungmarkendefinition entdeckt. Vergessen Sie nicht, daß hier alle Sprungmarken nur aus einem einzigen Großbuchstaben bestehen.
String zu lang (21): Eine Zeichenkette geht über das in AMOS-Basic zulässige Maximum von 65000 Zeichen hinaus.
Syntax Fehler in Animationskette (107): Es ist ein Fehler in der Animationssequenz, die Sie durch den Anim-Befehl spezifiziert haben. Überprüfen Sie ihn auf Tippfehler. Man setzt leicht einmal versehentlich einen Punkt "." statt eines Kommas ",".
Teilung durch Null (20): Sie haben versucht, eine Zahl durch Null zu teilen. Das ist in AMOS-Basic - und übrigens auch in allen anderen Basic-Programmen - nicht erlaubt.
Typenfehler (34): Einer Variable wurde ein Wert zugewiesen, der nicht zulässig ist. Zum Beispiel:

```
A$=12 Berichtigt: A$="12"
```

Überlauf (29): Das Ergebnis einer Berechnung übersteigt die maximale Größe einer Variablen.
Umrahmtes Fenster nicht am Rand des Schirms (59): Sie können ein umrahmtes Fenster nicht an den Rand des Bildschirms stellen. Sie müssen mindestens 8 Pixel zwischen dem Fenster und dem Bildschirm lassen, so daß Platz für den Rahmen bleibt.
Verzeichnis nicht gefunden (80): Das gewünschte Verzeichnis kann auf der aktuellen Diskette nicht gefunden werden. Sie haben vielleicht versehentlich die falsche Diskette eingelegt.
Verzeichnis nicht leer (85): Mit KILL kann man nur LEERE Verzeichnisse löschen.
Zu viele Farben in FLASH Sequenz (51): Sie haben die maximale Anzahl von 16 Farbveränderungen in einem einzigen FLASH-Befehl überschritten.

Fehlermeldungen für Erweiterungen

Die Befehle in den AMOS-Erweiterungsdateien rufen keine numerierten Fehlermeldungen hervor, Sie müssen jedoch trotzdem wissen, was sie bedeuten. Hier sind die Fehlermeldungen für die MUSIC-Erweiterung:

256 Werte fur eine Welle noetig: Wellen können durch eine Liste von 256 werten erzeugt werden.

Das ist kein Tracker Module: Sie haben versucht mit Track Play eine Bank adzuspielen, die nicht im Tracker Format vorlag.

Kann Narrator-Datel nicht oeffnen: AMOS kann die Library-Dateien der Systemdiskette nicht finden, die zum Laden des Narrator-Programms erfordelich sind.

Musik Bank nicht gefunden: Bevor Sie eine Musik abspielen können, müssen Sie diese mit Load einladen.

Musik nicht definiert: Die von Ihnen angesproche Musik wurde noch nicht zuvor definiert.

Sample nicht definiert: Sie haben versuchet, ein Sample zu spielen, das in der aktuellen Sample-Bank nicht vorhanden ist.

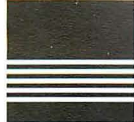
Sample bank nicht gefunden: Es ist keine Sample-Bank im Specicher.

Wellen 0 und 1 reserviert: Diese beiden Wellen werden von AMOS für die Befehle BELL und NOISE reserviert und können deshalb von Ihnen nicht verandert werden.

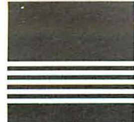
Welle nicht definiert: Die angesprochene Wellenform existiert gar nicht! Sie müssen sie zunachst mit Set Wave definieren.

Durch die Pack-Erweiterung können die folgenden beiden Fehlermeldungen hervorgerufen werden:

Keine gepackte Bitmap: Sie haben versucht, eine Datenbank zu entpacken, die nicht im Bitmap-Format ist.
Kein gepackter Schirm: Die Daten, die Sie entpacken möchten, sind nicht im gepackten Schirm-Format.



Command Index



Index

europress
SOFTWARE